การใช้ระบบเชิงวัตถุกับระบบงานบริหารพัสดุสายช่างอากาศ

OBJECT-ORIENTED SYSTEM FOR AERONAUTICAL SUPPLY MANGEMENT

นาวาอากาศโทหญิงนพพร สงวนทรัพย์

WING COMMANDER NOPPORN SA-NGUANSUP

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต

สาขาวิชาวิทยาการคอมพิวเตอร์และเทคโนโลยีสารสนเทศ

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2539

ISBN 974-621-587-6

OBJECT-ORIENTED SYSTEM FOR AERONAUTICAL SUPPLY MANGEMENTIGN

NOPPORN SA-NGUANSUP

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE

MASTER OF SCIENCE PROGRAM IN COMPUTER SCIENCE

AND INFORMATION TECHNOLOGY

SCHOOL OF GRADUATE STUDIES

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

1996

ISBN 974-621-587-6

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองวิทยานิพนธ์

หัวข้อวิทยานิพนธ์ การใช้ระบบเชิงวัตถุกับระบบงานบริหารพัสดุสายช่างอากาศ
OBJECT-ORIENTED SYSTEM FOR AERONAULTICAL SUPPLY MANAGEMENT

ชื่อนักศึกษา น.ท.หญิงนพพร สงวนทรัพย์ **รหัสประจำตัว** 34628016

หลักสูตร วิทยาศาสตรมหาบัณฑิต

สาขาวิชา วิทยาการคอมพิวเตอร์และเทคโนโลยีสารสนเทศ

ภาควิชา คณิตศาสตร์และวิทยาการคอมพิวเตอร์

อาจารย์ผู้ควบคุมวิทยานิพนธ์ ดร.ศุภมิตร จิตตะยโศธร

คณะกรรมการสอบวิทยานิพนธ์	ลายมือชื่อ
ผศ.ดร.ศุภมิตร จิตตะยโศธร	
พ.อ.ดร.วิชิต สาทรานนท์	
อาจารย์กฤตวัน เครือตราชู	
อาจารย์สุรสิทธิ์ วรรณไกรโรจน์	

ค่าระดับคะแนนที่เป็นเอกฉันท์จากคณะกรรมการสอบ GOOD

วัน/เดือน/ปี ที่สอบ 4 เมษายน 2539 เวลา 10.00 น. เป็นต้นไป

สถานที่สอบ สอบห้อง 234 ณ ห้องสอบสัมมนาชั้น 2 อาคารสำนักวิจัยและบริการคอมพิวเตอร์

บัณฑิตวิทยาลัยรับรองแล้ว

(รศ.ดร.มนัส สังวรศิลป์)

คณบดีบัณฑิตวิทยาลัย

วันที่.....๒.....**เดือน**.....มิถุนายน.....**พ.ศ**.....๒๕๓๙.....

หมายเหตุ การวัดผลวิทยานิพนธ์ให้ใช้ค่าระดับคะแนนดังนี้

ค่าระดับคะแนน	ผลการศึกษา	
O	Outstanding	(ดีเยี่ยม)
G	Good	(ดี)
P	Pass	(ผ่าน)
F	Fail	(ไม่ผ่าน)

หัวข้อวิทยานิพนธ์	การใช้ระบบเชิงวัตถุกับระบบงานบริหารงานพัสดุสายช่างอากาศ
นักศึกษา	น.ท.หญิง นพพร สงวนทรัพย์
อาจารย์ผู้ควบคุมวิทยานิพนธ์	ผศ.ดร.ศุภมิตร จิตตะยโศธร
ระดับการศึกษา	วิทยาศาสตรมหาบัณฑิต
	สาขาวิชาวิทยาการคอมพิวเตอร์และเทคโนโลยีสารสนเทศ
ภาควิชา	คณิตศาสตร์และวิทยาการคอมพิวเตอร์
	คณะวิทยาศาสตร์
	สถาบันเทคโนโลโยพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ.	2539

บทคัดย่อ

ซอฟท์แวร์ในปัจจุบันมีการเพิ่มคุณสมบัติที่ทำให้สามารถเขียนโปรแกรมแบบเชิงวัตถุได้ ตัวอย่างเช่น Visual Basic, Visual C, Delphi เป็นต้น ดังนั้นเพื่อเป็นการเตรียมการรองรับสำหรับซอฟท์แวร์ใหม่ๆ ในอนาคตที่มีแนวโน้มเปลี่ยนไปเป็นแบบเชิงวัตถุจึงได้นำเอาหลักการเชิงวัตถุมาออกแบบระบบงานบริหารพัสดุสายช่างอากาศของกรมช่างอากาศ ซึ่งหลักการคือกำหนดข้อมูลในระบบงานเป็นออบเจคคลาสแทนการกำหนดเป็นฐานข้อมูลเช่นในวิธีการออกแบบระบบแบบเดิม ภายในออบเจคคลาสประกอบด้วยข้อมูลและเมธธอด(หรือโพรซิเยอร์)รวมอยู่ด้วยกันในขณะที่แบบเก่าจะแยกส่วนของข้อมูลและโพรซิเยอร์(หรือโปรแกรมระบบนั่นเอง)ออกจากกัน ออกแบบระบบงานโดยใช้ออบเจคโมเดลลิ่งเทคนิคหรือโอเอ็มที (OMT : Object Modeling Technique) และพัฒนาระบบงานบนสมอลทอค (Smalltalk) สาเหตุที่เลือกสมอลทอคเพราะว่าสมอลทอคเป็นต้นแบบของซอฟท์แวร์เชิงวัตถุ

Thesis Title	Object-Oriented System for Aeronautical Supply Management
Student	Wg.Cdr. Nopporn Sa-nuangsup
Thesis Advisor	Asst.Prof. Dr. Supamit Jittayasothorn
Level of Study	Master of Science Program in
	Computer Science and Information Technology
Department	Mathematics and Computer Science
	Faculty of Science
	King Mongkut's Institute of Technology Ladkrabang
Year	1996

ABSTRACT

Today application developing tools such as Visual Basic, Visual C, Delphi are object-oriented. Applications implemented by these tools should therefore be object-oriented designed. In this thesis Aeronautical Enginering Depot supply management system is designed using Object Modeling Technique and implemented in Smalltalk. Smalltalk is chosen because object-oriented features are fully implemented and supported by the language.

กิตติกรรมประกาศ

วิทยานิพนธ์นี้ประสบความสำเร็จลงได้ด้วยดีก็เพราะว่าได้รับความกรุณาจากท่านผู้ช่วยศาสตราจารย์ ดร.ศุภมิตร จิตตะยโศธร ท่านกรุณาให้ข้อเสนอแนะและเป็นกำลังใจผลักดันจนกระทั่งทำให้ดิฉันได้มีวันนี้ ดังนั้นดิฉันจึงขอกราบขอบพระคุณท่านอาจารย์เป็นอย่างสูงไว้

ขอขอบพระคุณผู้บังคับบัญชาทุกท่าน เพื่อนทุกคน รวมทั้งเจ้าหน้าที่ของสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังทุกคน ที่ได้มีส่วนช่วยสนับสนุนผลงานวิทยานิพนธ์นี้

ท้ายที่สุดขอขอบคุณ นอ.ธวัช สงวนทรัพย์ ที่เสียสละให้ดิฉันใช้เวลาของครอบครัวทำงานวิทยานิพนธ์จนประสบความสำเร็จ

นพพร สงวนทรัพย์

สารบัญ

หน้า

สารบัญ (ต่อ)

บทที่ หน้า

สารบัญ (ต่อ)

บทที่ หน้า

สารบัญ (ต่อ)

บทที่ หน้า

สารบัญภาพ

หน้า

สารบัญภาพ (ต่อ)

หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและทฤษฎีที่ใช้

งานวิจัยนี้ได้นำเสนอการออกแบบระบบงานโดยใช้หลักการเชิงวัตถุ ซึ่งแตกต่างไปจากการออกแบบระบบงานแบบเดิม (Conventional) ที่ใช้วิธีการเก็บรวบรวมข้อมูลต่างๆ เป็นแฟ้มข้อมูล(File) หรือฐานข้อมูล (Database) แยกกันกับโปรแกรมระบบงาน ในขณะที่หลักการเชิงวัตถุ [1] จะกำหนดสิ่งต่างๆที่อยู่ภายในขอบเขตของระบบงานเป็นออบเจค (Object) ซึ่งในออบเจคจะประกอบด้วยข้อมูล (Data) และเมธธอด (Method) รวมกันอยู่ภายในออบเจค (Encapsulation) เมธธอดเปรียบเทียบได้ทำนองเดียวกันกับโพรซีเยอร์ (Procedure) ในโปรแกรมระบบงานแบบเดิมนั่นเอง การเรียกใช้เมธธอดหรือการสั่งให้เกิดการกระทำตามที่กำหนดไว้ในเมธธอดคือการส่งเมสเซส(Message) ไปยังออบเจคนั้นๆ ดังนั้นหากมีการเปลี่ยนแปลงวิธีการกระทำ (Function) ใดๆ ของ ออบเจคจะไม่มีผลกระทบกับเมสเซสที่ส่งไปยังออบเจคนั้นและจะไม่มีผลกระทบกับออบเจคอื่นๆด้วย ซึ่งนับว่าเป็นผลดีอย่างมากกับระบบงานเพราะโดยปกติแล้วพบว่าระบบงานมักจะมีการเปลี่ยนแปลงวิธีการปฏิบัติอยู่เสมอๆ เมธธอดสามารถใช้อธิบายถึงพฤติการณ์ของออบเจคแต่ละออบเจคได้ ออบเจคที่มีพฤติการณ์แบบเดียวกันสามารถจัดรวมกลุ่มเป็นกลุ่มเดียวกันได้เรียกว่าคลาส (Class)ในระหว่างคลาสสามารถที่จะถ่ายทอดคุณสมบัติ (Inheritance) จากคลาสหนึ่งไปยังอีกคลาสหนึ่งได้ โดยเรียกคลาสที่ถ่ายทอดคุณสมบัติว่าซุปเปอร์คลาส (Super class) และเรียกคลาสที่ได้รับการถ่ายทอดคุณสมบัติว่าซับคลาส (Subclass) ซึ่งทำให้การบำรุงรักษาระบบงานทำได้ง่าย เริ่มต้นงานวิจัยด้วยการศึกษาหลักการเชิงวัตถุต่อจากนั้นทำการวิเคราะห์เชิงวัตถุ (Object-Oriented Analysis) โดยใช้วิธีออบเจคโมเดลลิ่งเทคนิค (Object Modeling Technique) หรือโอเอ็มที (OMT) [2] ในระบบงานบริหารพัสดุสายช่างอากาศกำหนดคลาส Item เป็น คลาสสำหรับพัสดุทั้งหมด คลาส Item ประกอบด้วยสองซับคลาส คือคลาส Reparable และคลาส Consumable โดยคลาส Reparable คือคลาสของพัสดุประเภทพัสดุซ่อมหมุนเวียนและคลาส Consumable คือคลาสของพัสดุประเภทพัสดุสิ้นเปลือง หลังจากนั้นสร้างแบบจำลองเพื่อใช้อธิบายระบบงานทั้งระบบงานซึ่งมีอยู่ด้วยกัน 3 แบบจำลอง คือหนึ่งออบเจคโมเดล (Object Model) ใช้อธิบายรายละเอียดเกี่ยวกับออบเจคนั้นว่ามีข้อมูลและเมธธอดอะไรบ้างและแสดงความสัมพันธ์ระหว่างออบเจคแต่ละออบเจคภายในระบบงาน แบบจำลองที่สองคือไดนามิกโมเดล (Dynamic Model) ซึ่งใช้สเตดไดอะแกรม (State Diagram) อธิบายพฤติกรรมของออบ

เจคที่กระทำโต้ตอบกันภายในระบบ ไดนามิกโมเดลนี้ไม่ค่อยมีความจำเป็นมากนักสำหรับระบบงานบริหารพัสดุสายช่างอากาศ เพราะว่าเป็นระบบงานที่เกี่ยวกับการเก็บข้อมูลเสียเป็นส่วนใหญ่ แบบจำลองสุดท้ายคือฟังก์ชั่นนัลโมเดล(Functional Model) ซึ่งใช้ดาต้าโฟลไดอะแกรม (Data Flow Diagram) อธิบายถึงวิธีการคำนวณของเมธธอดภายในออบเจคแต่ละออบเจค หลังจากสร้างแบบจำลองเสร็จแล้วอันดับต่อไปก็คือการพัฒนาระบบงาน (Implementation) บนสมอลทอค [3]

1.2 สาเหตุที่เลือกใช้ออบเจคโมเดลลิ่งเทคนิค

วิธีการเชิงวัตถุ (Object-Oriented Methodology) มีอยู่หลายวิธีด้วยกัน เช่น ออบเจคโอเรียนเตดอะนาไลซิส(Object-Oriented System Analysis) ซึ่งสร้างขึ้นโดยแชลเออร์และแมลเลอร์ (Shlaer and Mellor) [Shlaer 1988] [4] โดยใช้อีอาร์ไดอะแกรม(ER-diagram) อธิบายโครงสร้างของข้อมูล ต่อมาในปี 1990 Coad-Yourdon ได้สร้างออบเจคโอเรียนเตดอะนาไลซิส (Object-Oriented Analysis) หรือโอโอเอ (OOA) [5] ซึ่งคล้ายๆ กับออบ เจคโอเรียนเตดซิสเต็มอะนาไลซิส (Object-Oreinted System Analysis) หรือโอเอสเอ (OSA) ซึ่งพัฒนามาจาก ออบเจครีเลชันชิพโมเดล (Object Relationship Model) หรือโออาร์เอ็ม (ORM) [6] โออาร์เอ็มเป็นผลงานวิจัยของเคร์ก (Kurtz) [Kurtz 1988] โอโอเอแตกต่างกับโอเอสเอคือโอเอสเอนั้นสะท้อนถึงรูปแบบของการเขียนโปรแกรมเชิงวัตถุ (Object-Oriented Programming) ในปี 1991 เจมส์แรมบอก (James Rambaugh) สร้างออบ เจคโมเดลลิ่งเทคนิค (Object Modeling Technique-OMT) ซึ่งใช้เอ็กซ์เทนเดดอีอาร์ไดอะแกรม(extended ER diagram) อธิบายโครงสร้างของออบเจค ใช้สเตทไดอะแกรม (state diagram) อธิบายถึงพฤติกรรมโต้ตอบระหว่างออบเจค และใช้ดาต้าโฟลไดอะแกรม (data flow diagram) อธิบายวิธีการคำนวณของเมธธอดในออบเจค เราจะสังเกตุเห็นได้ว่าการวิเคราะห์เชิงวัตถุแต่ละวิธีจะคล้ายๆ กัน ดังนั้นผู้ทำวิจัยจึงเลือกใช้วิธีออบเจคโมเดลลิ่งเทคนิคในงานวิจัยนี้ การออกแบบระบบงานโดยใช้หลักการเชิงวัตถุเท่าที่ผ่านมามักจะอยู่ในรูปของปัญหาสั้นๆ หรือเป็นเพียงบางส่วนเท่านั้น ดังนั้นผู้วิจัยจึงคิดที่จะทดลองออกแบบระบบงานบริหารพัสดุสายช่างอากาศแบบครบวงจร (Real Lift System) โดยใช้ออบเจคโมเดลลิ่งเทคนิค และพัฒนาระบบงานบนสมอลทอค เพราะว่าสมอลทอคเป็นต้นแบบของซอฟต์แวร์เชิงวัตถุ

1.3 โครงสร้างวิทยานิพนธ์

รายละเอียดของเนื้อหาในแต่ละบทถัดไปมีดังนี้

บทที่ 2 ออบเจคโมเดลลิ่งเทคนิคและสมอลทอค กล่าวถึงหลักการของออบเจคโมเดลลิ่งเทคนิคและโปรแกรมภาษาสมอลทอค

บทที่ 3 ระบบงานบริหารพัสดุสายช่างอากาศ กล่าวถึงลักษณะของระบบงานซึ่งจะใช้ศึกษาในงานวิจัยนี้

บทที่ 4 การพัฒนาระบบงาน กล่าวถึงวิธีการออกแบบโดยใช้วิธีของออบเจคโมเดลลิ่งเทคนิค และการพัฒนาระบบงานที่ใช้ศึกษาโดยใช้โปรแกรมสมอลทอค

บทที่ 5 การซ่อมบำรุงระบบเชิงวัตถุ กล่าวถึงการซ่อมบำรุงระบบเชิงวัตถุของสมอลทอค

บทที่ 6 สรุปและข้อเสนอแนะ กล่าวถึงผลสรุปของงานวิจัย และข้อเสนอแนะ

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 ออบเจคโมเดลลิ่งเทคนิค (Object Modeling Technique : OMT)

ออบเจคโมเดลลิ่งเทคนิคหรือโอเอ็มที ประกอบด้วยโมเดลที่ใช้ในการอธิบายระบบงานที่แตกต่างกัน 3 โมเดลด้วยกันคือ

1. ออบเจคโมเดล (Object Model)
2. ไดนามิกโมเดล (Dynamic Model)
3. ฟังก์ชันนัลโมเดล (Functional Model)

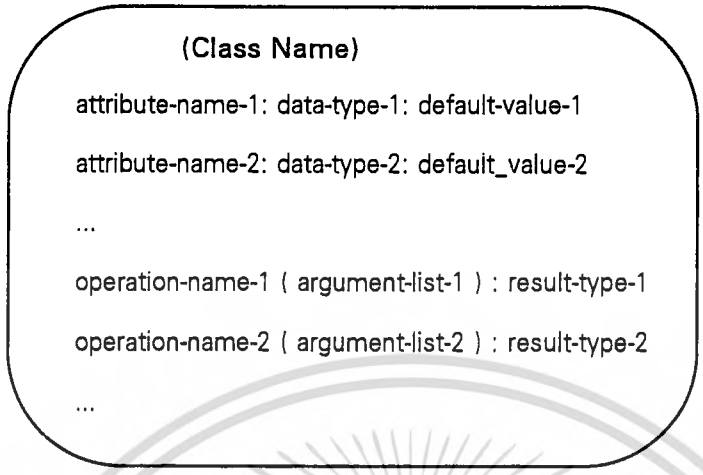
2.1.1 ออบเจคโมเดล

ออบเจคโมเดลเป็นโมเดลที่สำคัญที่สุดในจำนวนโมเดลทั้งสามโมเดล ออบเจคโมเดลอธิบายโครงสร้างที่ไม่มีการเปลี่ยนแปลงของระบบ โดยแสดงออบเจคต่างๆ ภายในระบบ ความสัมพันธ์ระหว่างออบเจค (Association) แอทริบิวท์ (Attribute) และโอเปอเรชัน (Operation) ทั้งหมดของแต่ละออบเจค

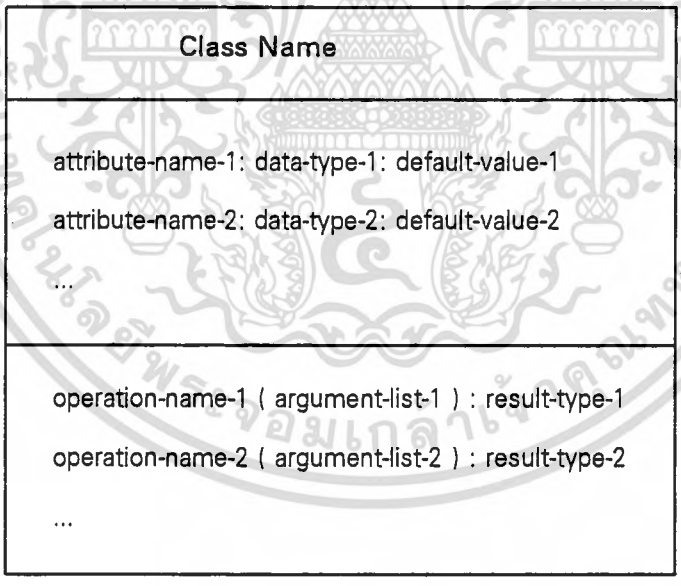
2.1.1.1 ออบเจคไดอะแกรม (Object diagram)

แผนผังที่ใช้แทนออบเจคคือออบเจคไดอะแกรม โดยใช้สัญญลักษณ์รูปสี่เหลี่ยมผืนผ้ามุมมน ภายในรูปสี่เหลี่ยมผืนผ้ามุมมนเขียนชื่อของคลาสด้วยตัวอักษรทึบและอยู่ภายในเครื่องหมายวงเล็บ บรรทัดถัดมาเป็นชื่อของออบเจค แอทริบิวท์ และโอเปอเรชันทั้งหมดของออบเจค ดังแสดงในภาพที่ 1

แผนผังที่ใช้แทนคลาสคือรูปสี่เหลี่ยมผืนผ้า ภายในรูปสี่เหลี่ยมผืนผ้าเขียนชื่อของคลาสด้วยตัวอักษรทึบ บรรทัดถัดมาเป็นแอทริบิวท์ และโอเปอเรชันทั้งหมดของออบเจค โดยระหว่างชื่อคลาส แอทริบิวท์ และโอเปอเรชันจะมีเส้นตรงแบ่งคั่นเพื่อให้แผนผังของคลาสแตกต่างไปจากแผนผังของออบเจค ดังแสดงตัวอย่างในภาพที่ 2



ภาพที่ 1 ออบเจคไดอะแกรม



ภาพที่ 2 คลาสไดอะแกรม

2.1.1.2 แอสโซซิเอชัน (Association)

สัญญลักษณ์ของแอสโซซิเอชันคือเส้นตรงที่ลากต่อระหว่างคลาส ซึ่งใช้แสดงความสัมพันธ์ของออบเจคภายในคลาส มัลติพลิซิตี้ (Multiplicity) คือการกำหนดจำนวนสมาชิกของคลาสหนึ่งซึ่งมีความสัมพันธ์กับสมาชิกของอีกคลาสหนึ่ง โดยวิธีเขียนสัญญลักษณ์บนแอสโซซิเอชันชิดกับคลาสนั้น โดยปกติมักใช้คำว่า one หรือ many

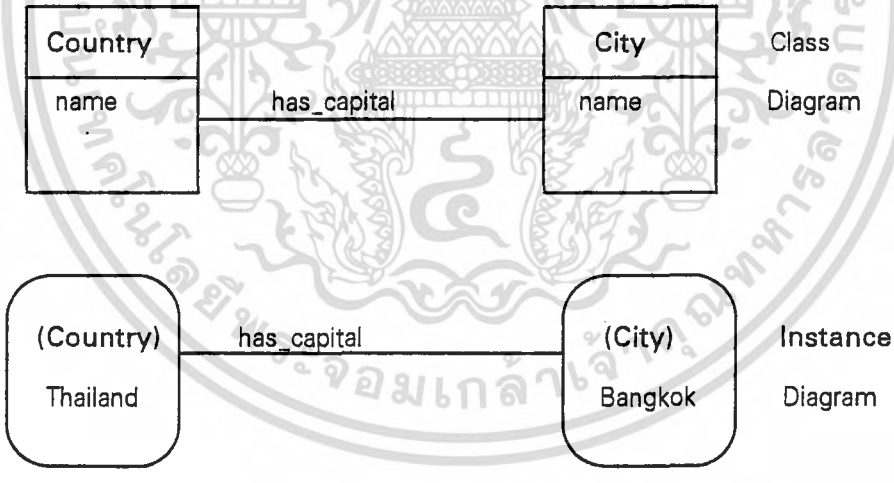
สัญญลักษณ์ 1+ หมายถึงหนึ่งหรือมากกว่า

2,4 หมายถึง 2 และ 4

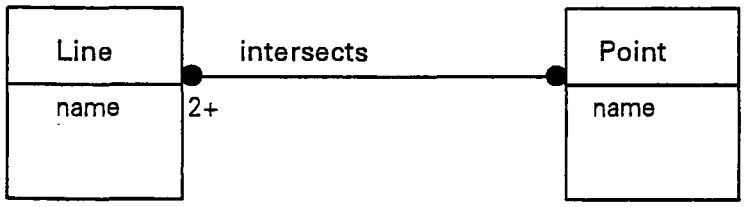
○— หมายถึงศูนย์หรือหนึ่ง

●— หมายถึงศูนย์หรือมากกว่าศูนย์

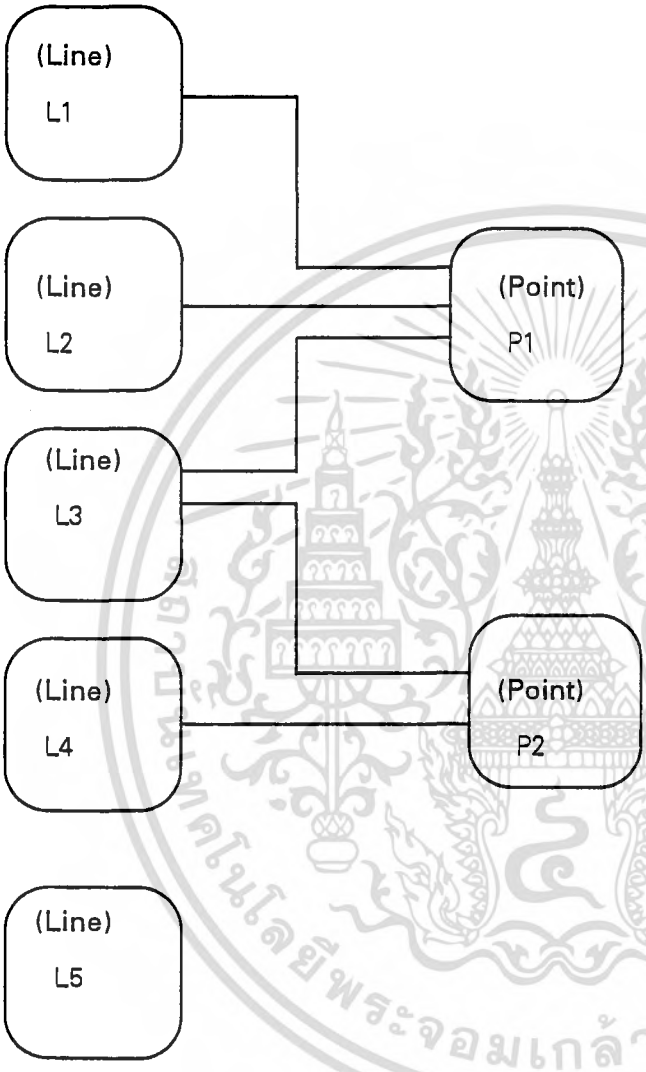
ดังตัวอย่างในภาพที่ 3 ถึงภาพที่ 5 ในภาพที่ 3 แสดงความสัมพันธ์แบบ หนึ่งต่อหนึ่ง นั่นคือประเทศหนึ่งมีเมืองหลวงได้เพียงหนึ่งเมืองเท่านั้น ภาพที่ 4 แสดงความสัมพันธ์แบบ มากต่อมาก นั่นคือจุด ๆ หนึ่งมีเส้นตรงตัดผ่านมากกว่าหนึ่งเส้น และเส้นตรงเส้นหนึ่งผ่านจุดตัดมากกว่าหนึ่งจุดหรือไม่ผ่านจุดใดเลย ภาพที่ 5 แสดงความสัมพันธ์แบบ Ternary ใช้สัญญลักษณ์รูปสี่เหลี่ยมขนมเปียกปูนเป็นจุดเชื่อมต่อแอสโซซิเอชัน



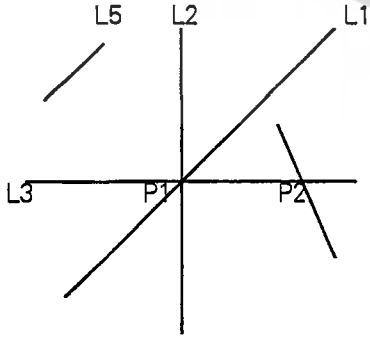
ภาพที่ 3 ตัวอย่างความสัมพันธ์แบบ หนึ่ง-ต่อ-หนึ่ง (one-to-one association)



Class Diagram

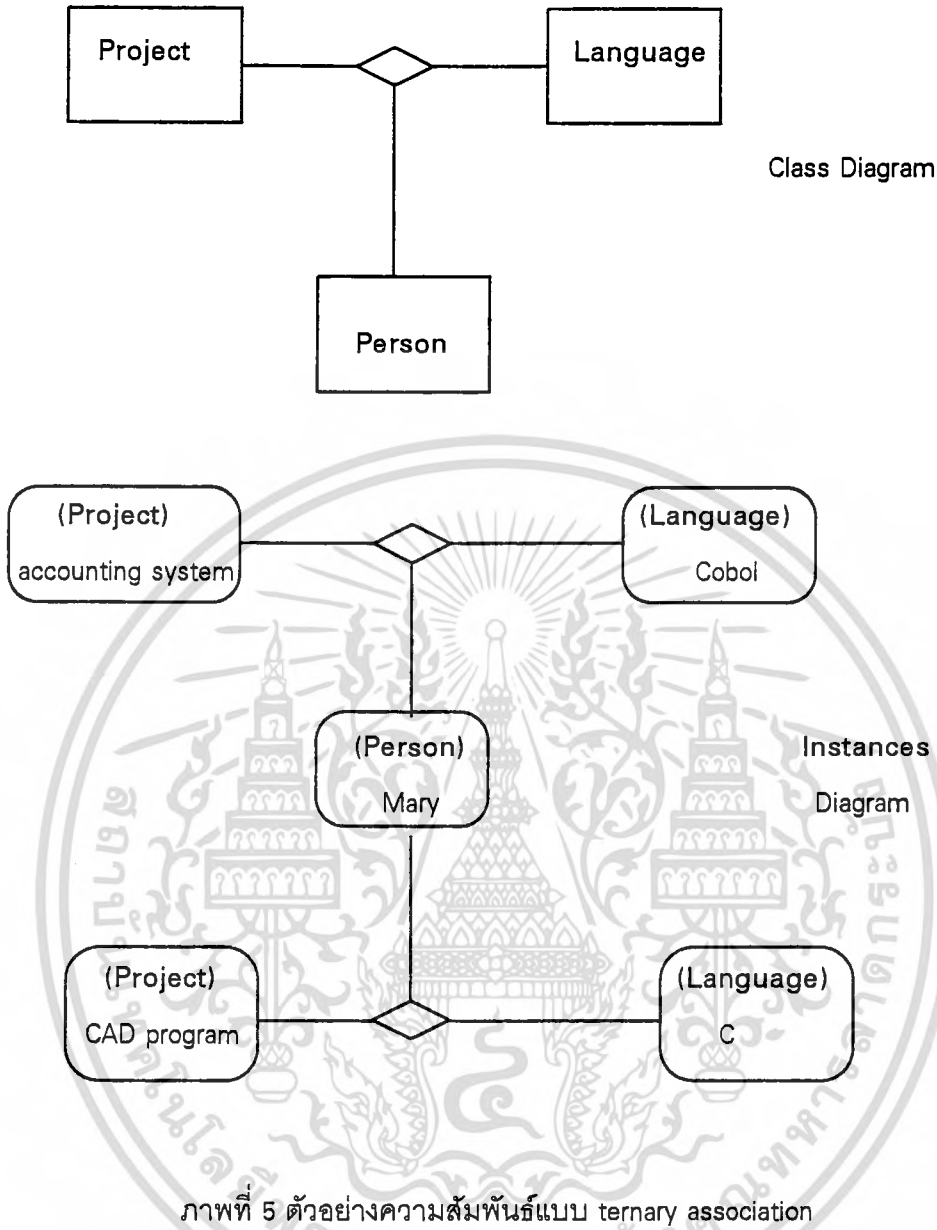


Instances Diagram



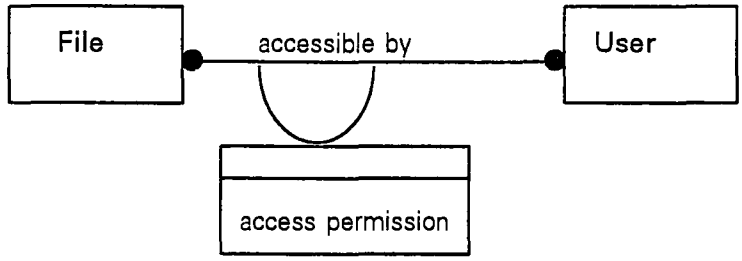
Sample Data

ภาพที่ 4 ตัวอย่างความสัมพันธ์แบบ many-to-many association

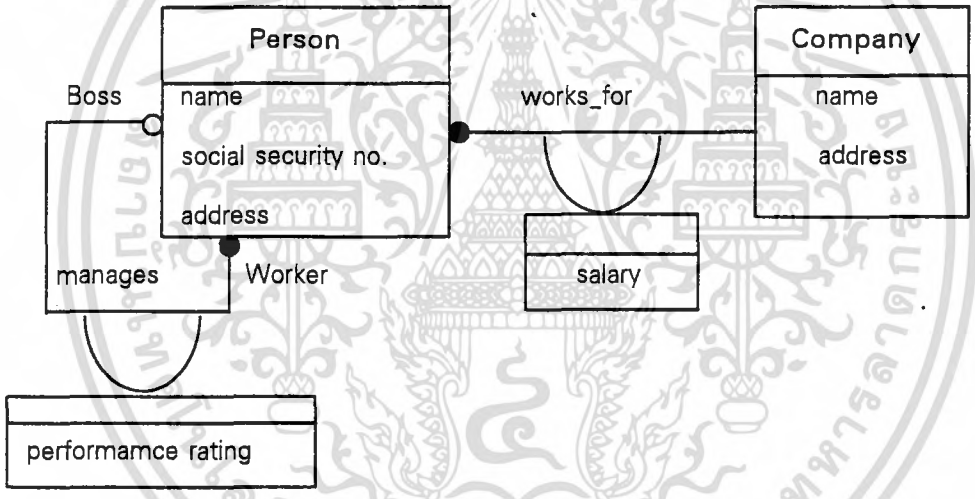


ภาพที่ 5 ตัวอย่างความสัมพันธ์แบบ ternary association

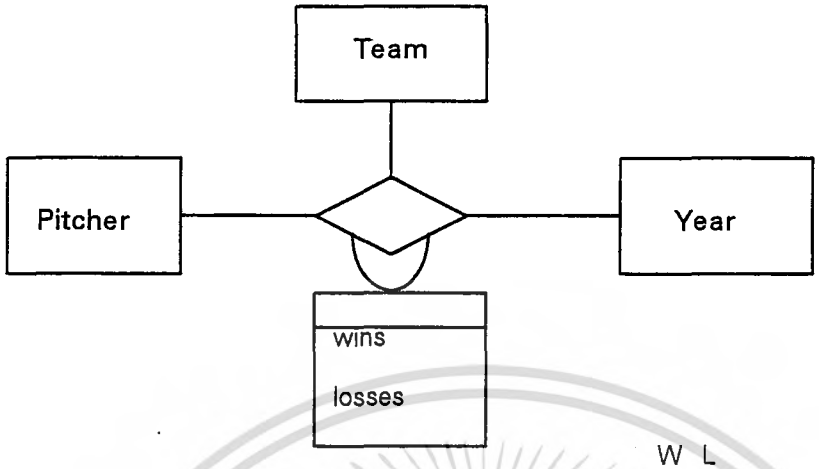
ลิงค์แอทริบิวท์ (Link Attribute) คือคุณสมบัติของลิงค์ในแอสโซซิเอชัน ในทำนองเดียวกับแอทริบิวท์คือคุณสมบัติของออบเจคในคลาส สัญญลักษณ์ของลิงค์แอทริบิวท์คือรูปสี่เหลี่ยมสัมผัสกับแอสโซซิเอชันด้วยเส้นโค้งที่ลากต่อระหว่างรูปสี่เหลี่ยมกับแอสโซซิเอชันดังตัวอย่างในภาพที่ 6 ถึงภาพที่ 8



ภาพที่ 6 ลิงค์แอทริบิวท์สำหรับความสัมพันธ์แบบ many-to-many



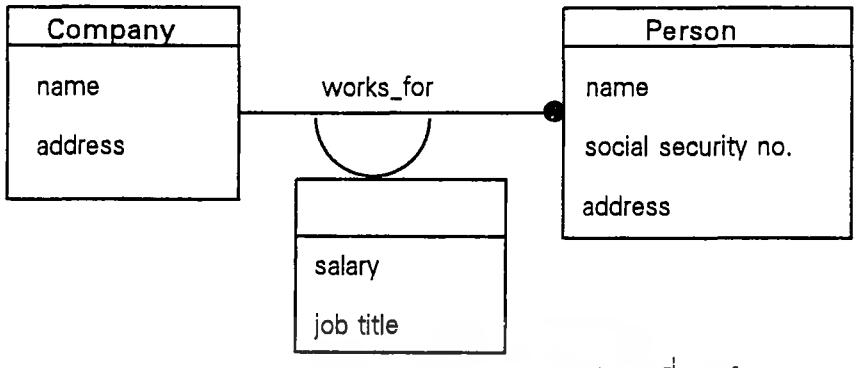
ภาพที่ 7 ลิงค์แอทริบิวท์สำหรับความสัมพันธ์แบบ one-to-many



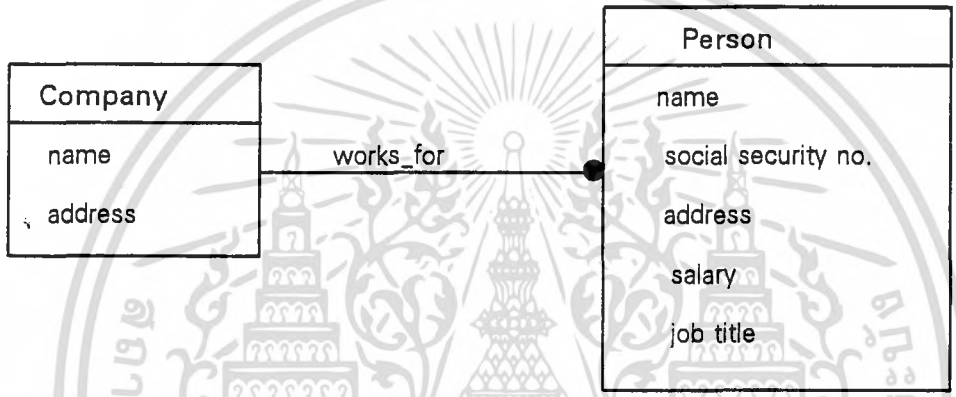
			W	L
Harry Eisenstat	Cleveland Indians	1939	6	7
Harry Eisenstat	Detroit Tigers	1939	2	2
Willis Hudin	Cleveland Indians	1939	9	10
Willis Hudin	Cleveland Indians	1940	2	1

ภาพที่ 8 ลิงค์แอทริบิวท์สำหรับความสัมพันธ์แบบ Ternary

ในกรณีของ one-to-one และ one-to-many ลิงค์แอทริบิวท์อาจจะสามารถรวมเข้าไว้ในคลาสได้ เช่นตัวอย่างในภาพที่9 แต่ในกรณีของ many-to-many ไม่สามารถจะกระทำได้ โดยกฎแล้วไม่ควรรวมลิงค์แอทริบิวท์เข้าไว้ในคลาส เพราะจะทำให้ความยืดหยุ่นลดลงถ้ามัลติพลิซิตี้ของ Works-for เปลี่ยนแปลง เช่นในตัวอย่างภาพที่9 ถ้า Works-for เปลี่ยนเป็น many-to-many รูปแบบที่เป็นลิงค์แอทริบิวท์เท่านั้นที่ถูกต้อง



รูปแบบที่ควรทำ



รูปแบบที่ไม่ควรทำ

ภาพที่ 9 ลิงค์แอทริบิวท์เปรียบเทียบกับออบเจคแอทริบิวท์

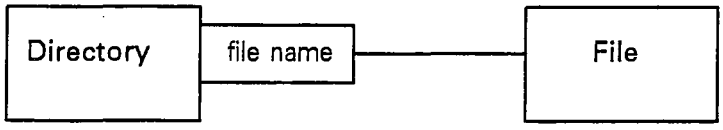
ชื่อของบทบาท (Role Name) ใช้อธิบายบทบาทของคลาสในแอสโซซิเอชันนั้น โดยเขียนชื่อของบทบาทที่เหนือแอสโซซิเอชันใกล้ๆ กับคลาสที่แสดงบทบาท เช่นตัวอย่างในภาพที่ 7

ควอลลิฟิเคชัน (Qualification) คือแอทริบิวท์พิเศษซึ่งลดมัลติพลิซิตี้ของแอสโซซิเอชันแบบ one-to-many และ many-to-many โดยใช้สัญญลักษณ์เป็นรูปกล่องสี่เหลี่ยมเล็กๆ ที่ปลายของแอสโซซิเอชันใกล้กับคลาสที่ถูกกำหนดควอลลิไฟเออร์ เช่น ตัวอย่างในภาพที่ 10

2.1.1.3 อะกรีเกชัน (Aggregation)

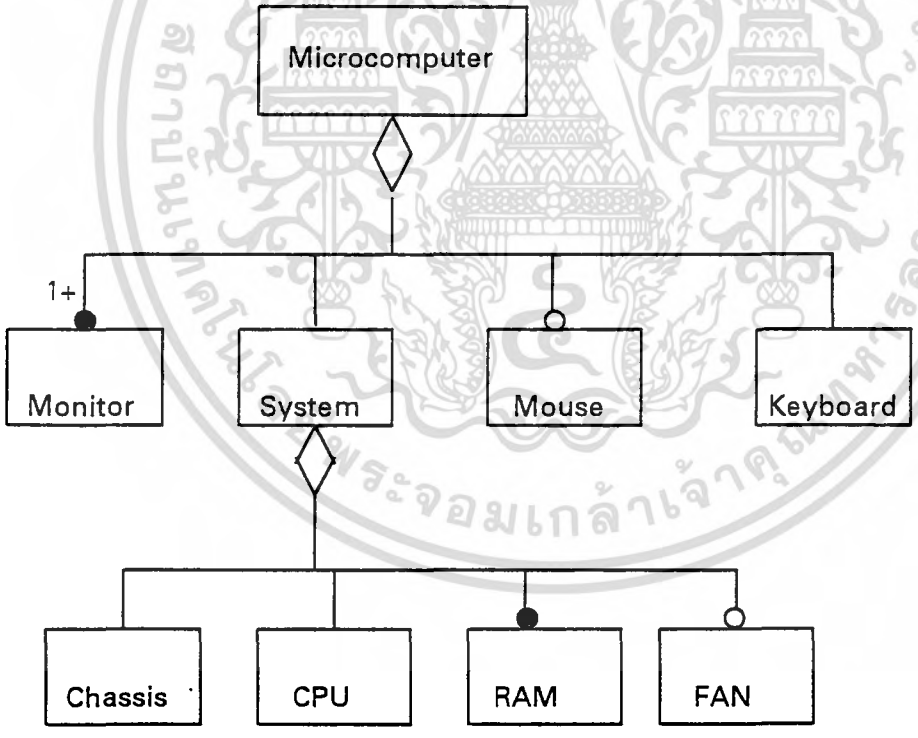
คือความสัมพันธ์แบบ "part-whole" หรือ "a-part-of" ซึ่งใช้อธิบายความสัมพันธ์ในรูปแบบที่ออบเจคหนึ่งเป็นส่วนประกอบของอีกออบเจคหนึ่ง อะกรีเกชันใช้สัญญลักษณ์เช่นเดียวกับ

แอสโซซิเอชันแต่มีสัญญลักษณ์สี่เหลี่ยมขนมเปียกปูนที่ปลายของแอสโซซิเอชันเพื่อระบุออบเจคที่เป็นทั้งหมด(whole) เช่น ตัวอย่างในภาพที่ 11 และภาพที่ 12



ภาพที่ 10 ตัวอย่างการกำหนดควอลลิไฟเออร์

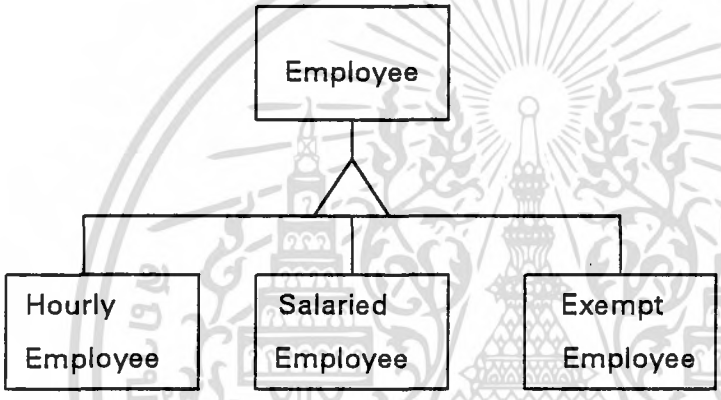
ภาพที่ 11 ตัวอย่างอะกรีเกชัน



ภาพที่ 12 อะกรีเกชันหลายระดับ

2.1.1.4 เจนเนอรัลไลเซชัน (Generalization)

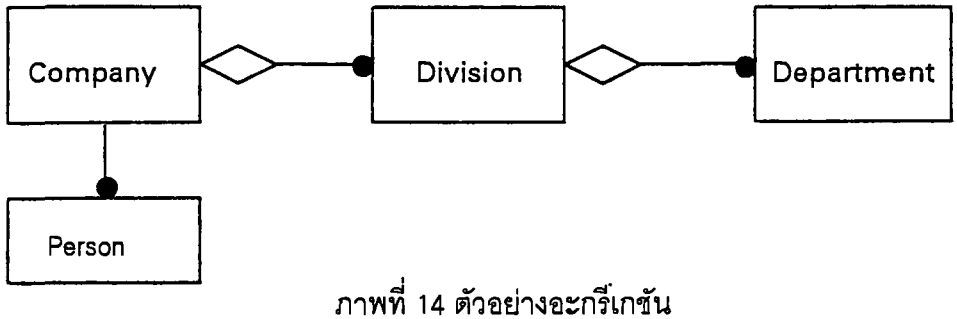
คือความสัมพันธ์ระหว่างคลาสซึ่งคลาสใหม่ได้มาจากการดัดแปลงคลาสที่มีอยู่แล้วโดยเรียกคลาสที่เป็นต้นฉบับว่าซุปเปอร์คลาส (Super Class) คลาสที่ดัดแปลงขึ้นมาใหม่เรียกว่าซับคลาส (Subclass) สัญญลักษณ์ที่ใช้ในแอสโซซิเอชันคือรูปสามเหลี่ยม โดยยอดของสามเหลี่ยมชี้ไปที่ซุปเปอร์คลาสและฐานของสามเหลี่ยมสัมผัสกับแอสโซซิเอชันซึ่งติดต่อกับซับคลาส เช่นตัวอย่างในภาพที่14 เจนเนอรัลไลเซชันนี้บางทีเรียกว่าความสัมพันธ์แบบ "a-kind-of" หรือ "is-a" แต่ละซับคลาสจะถ่ายทอดลักษณะทั้งหมดของซุปเปอร์คลาส และเพิ่มแอทริบิวท์, โอเปอเรชันที่เป็นลักษณะเฉพาะของซับคลาสนั้นเข้าไป



ภาพที่ 13 ตัวอย่างเจนเนอรัลไลเซชัน

2.1.1.5 เปรียบเทียบอะกรีเกชันกับแอสโซซิเอชัน

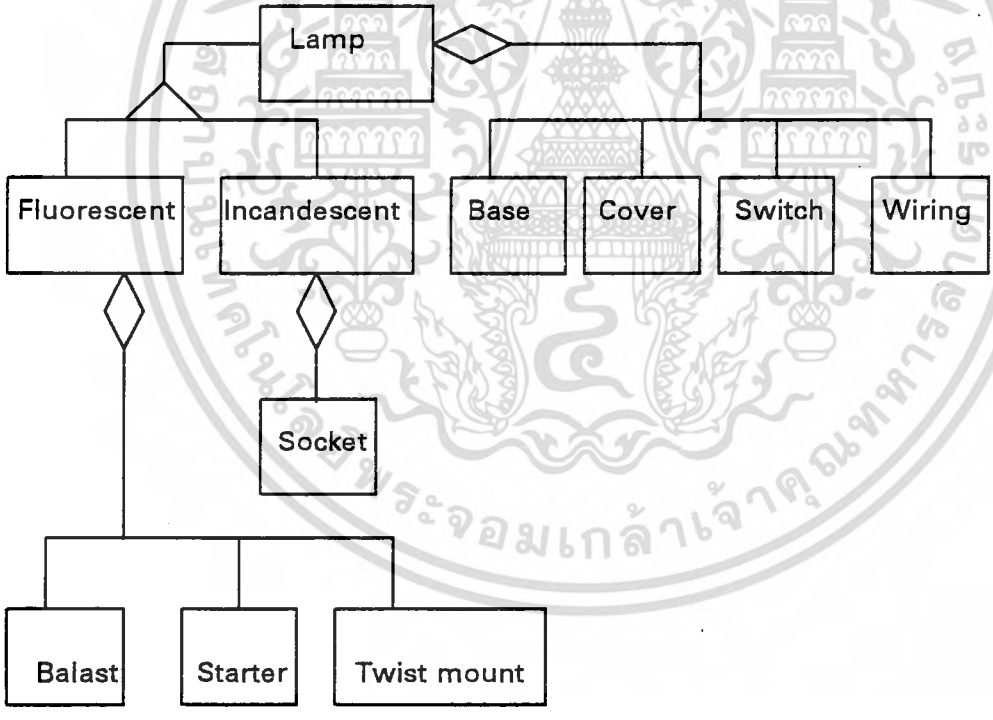
อะกรีเกชันคือรูปแบบพิเศษของแอสโซซิเอชัน ซึ่งมีแนวความคิดที่ไม่ได้แยกออบเจคออกจากกันอย่างเป็นอิสระถ้าออบเจคสองออบเจคมีความสัมพันธ์กันแบบออบเจคหนึ่งเป็นส่วนประกอบของอีกออบเจคหนึ่งซึ่งโดยปกติแล้วออบเจคทั้งสองออบเจคสามารถที่จะแยกออกจากกันเป็นสองออบเจคที่เป็นอิสระต่อกันได้ อะกรีเกชันสามารถอธิบายความสัมพันธ์ในกรณีดังกล่าวนี้ได้ เช่น ตัวอย่างในภาพที่ 15



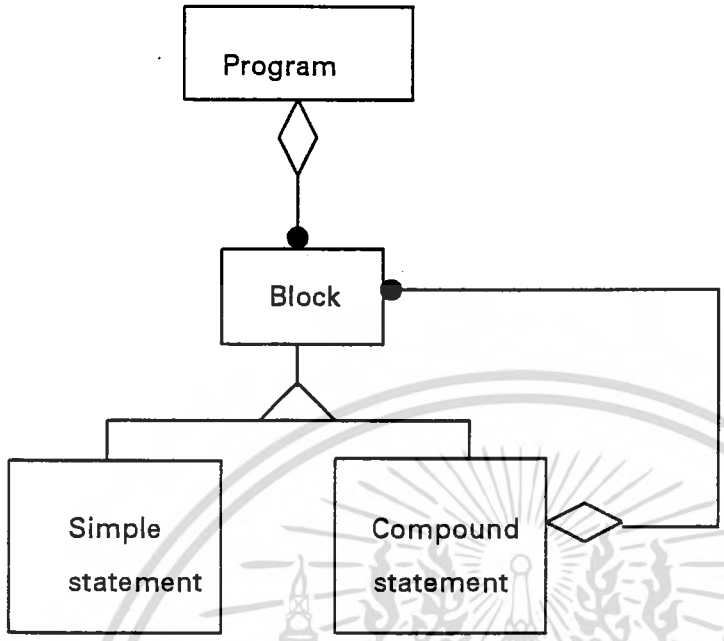
ภาพที่ 14 ตัวอย่างอะกรีเกชัน

2.1.1.6 เปรียบเทียบอะกรีเกชันกับเจนเนอรัลไลเซชัน

อะกรีเกชันมักจะอ้างถึงออบเจคที่แตกต่างกันแต่มีความสัมพันธ์กันในแง่ที่ออบเจคหนึ่งเป็นส่วนประกอบของอีกออบเจคหนึ่ง ส่วนเจนเนอรัลไลเซชันมักจะอ้างถึงคลาสโดยออบเจคที่เป็นสมาชิกของซับคลาสจะเป็นสมาชิกของซุปเปอร์คลาสด้วยเสมอ ดังนั้นบางครั้งจึงเรียกความสัมพันธ์แบบอะกรีเกชันว่า "and" และเรียกความสัมพันธ์แบบเจนเนอรัลไลเซชันว่า "or" อะกรีเกชันสามารถเป็นได้ทั้งแบบอะกรีเกชันคงที่ (Fixed aggregation), อะกรีเกชันเปลี่ยนแปลงได้ (Variable aggregation), หรือรีเคอซีพอะกรีเกชัน (Recursive aggregation) ตัวอย่างของ Lamp ในภาพที่1 15 คืออะกรีเกชันแบบคงที่ ตัวอย่างของ Company ในภาพที่ 15 คืออะกรีเกชันแบบเปลี่ยนแปลงได้ เพราะบริษัทมีได้หลายฝ่ายและแต่ละฝ่ายมีได้หลายแผนก ตัวอย่าง ของรีเคอซีพอะกรีเกชันคือ สมาชิกของอะกรีเกชันมีชนิดเป็นแบบเดียวกันกับอะกรีเกชันนั้น ตัวอย่างเช่นโปรแกรมคอมพิวเตอร์ในภาพที่ 16



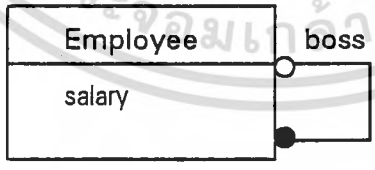
ภาพที่ 15 ตัวอย่างอะกรีเกชันและเจนเนอรัลไลเซชัน



ภาพที่ 16 รีเคอร์ซีพอะกรีเกชัน

2.1.1.7 **ข้อบังคับ** (Constraints)

คือข้อกำหนดการกระทำระหว่างเอ็นทิติ (Entities) ในออบเจคโมเดล เอ็นทิติหมายถึง ออบเจคคลาส แอทริบิวท์ ลิงค์ (Link) และแอสโซซิเอชัน ข้อบังคับจะกำหนดค่าที่เอ็นทิติสามารถจะมีได้ ตัวอย่างเช่น ไม่มีพนักงานคนไหนมีเงินเดือนมากกว่าหัวหน้า(ข้อบังคับระหว่างสองสิ่งในเวลาเดียวกัน) ดังแสดงในภาพที่ 17



ภาพที่ 17 ตัวอย่างของข้อบังคับ

2.1.2 ไดนามิกโมเดล

โครงสร้างแบบคงที่ทำให้เราสามารถเข้าใจระบบได้ง่ายที่สุดเป็นอันดับแรก ซึ่งโครงสร้างนั้นก็คือโครงสร้างของออบเจคและความสัมพันธ์ระหว่างออบเจค ในช่วงเวลาเดียวกันเราพบว่ามีการเปลี่ยนแปลงเกิดขึ้นกับออบเจคและความสัมพันธ์ระหว่างออบเจคตลอดเวลา ลักษณะของระบบซึ่งเกี่ยวข้องกับเวลาและการเปลี่ยนแปลงคือไดนามิกโมเดล ซึ่งตรงกันข้ามกับความคงที่หรือออบเจคโมเดล ไดนามิกโมเดลอธิบายลำดับของโอเปอเรชันที่เกิดขึ้นเพื่อตอบสนองต่อสิ่งเร้าจากภายนอกระบบโดยไม่คำนึงถึงว่าโอเปอเรชันนั้นทำอะไร ทำกับอะไร หรือโอเปอเรชันถูกสร้างขึ้นมาอย่างไร แนวความคิดหลักของไดนามิกโมเดลคืออีเวนท์ (Event) ซึ่งแทนสิ่งเร้าจากภายนอกระบบและสเตท (State) ซึ่งแทนค่าของออบเจค

ออบเจคโมเดลอธิบายรูปแบบที่เป็นไปได้ของออบเจค แอทริบิวท์ และลิงค์ซึ่งมีอยู่ในระบบ ค่าของแอทริบิวท์และลิงท์ที่อยู่ในออบเจคเรียกว่าสเตท ตลอดเวลาออบเจคมีการกระตุ้นซึ่งกันและกันผลลัพธ์คือลำดับของการเปลี่ยนแปลงสเตท สิ่งเร้าจากออบเจคหนึ่งถึงอีกออบเจคหนึ่งคืออีเวนท์ การตอบสนองอีเวนท์ขึ้นกับสเตทของออบเจคขณะกำลังรับอีเวนท์นั้น สเตทไดอะแกรมคือแผนผังเนตเวิร์คของสเตทและอีเวนท์ ในทำนองเดียวกับออบเจคไดอะแกรมคือแผนผังเนตเวิร์คของคลาสกับความสัมพันธ์ของคลาส ไดนามิกโมเดลประกอบด้วยสเตทไดอะแกรมหลาย สเตทไดอะแกรม โดยสเตทไดอะแกรมหนึ่งสำหรับคลาส ๆ หนึ่ง ซึ่งมีการเปลี่ยนแปลงพฤติกรรมที่สำคัญและแสดงกิจกรรมของระบบทั้งระบบ

2.1.2.1 อีเวนท์

อีเวนท์คือบางสิ่งบางอย่างที่เกิดขึ้น ณ เวลาหนึ่ง อีเวนท์ไม่มีช่วงเวลา ถึงแม้ว่าจะไม่มีอะไรเกิดขึ้นทันทีทันใดก็ตาม อีเวนท์หนึ่งอาจจะเกิดหลังอีเวนท์หนึ่งหรืออีเวนท์สองอีเวนท์อาจจะไม่มีความเกี่ยวข้องกันเลย เช่น เที่ยวบิน 123 ออกจากชิคาโกก่อนหรือหลังเที่ยวบิน 456 ถึงโรม เรียกสองอีเวนท์ที่ไม่เกี่ยวข้องกันนี้ว่าพ้องกัน (Concurrent) ซึ่งไม่มีผลกระทบซึ่งกันและกัน ในโมเดลของระบบจะต้องไม่เขียนลำดับของอีเวนท์เพราะว่ามันสามารถเกิดขึ้นในลำดับใดก็ได้ อีเวนท์เป็นวิธีหนึ่งของการส่งผ่านข้อมูลจากออบเจคหนึ่งไปยังออบเจคอื่นซึ่งไม่เหมือนกับการเรียกใช้ซับรูทีน (Subroutine) ที่มีการส่งค่ากลับคืน ในความเป็นจริงออบเจคมีอยู่พร้อมๆ กัน ออบเจคที่ส่งอีเวนท์ถึงอีกออบเจคหนึ่งอาจจะคาดหวังที่จะได้รับคำตอบ แต่คำตอบก็คืออีเวนท์อีกอีเวนท์หนึ่งที่แยกกันอยู่ภายใต้การควบคุมของออบเจคที่สองซึ่งอาจจะเลือกส่งหรือไม่ส่งคำตอบก็ได้ ทุกๆ อีเวนท์คือสิ่งที่เกิดขึ้นไม่ซ้ำกันเรารวมกลุ่มของอีเวนท์เป็นอีเวนท์คลาสและตั้งชื่อเพื่อระบุ

โครงสร้างและพฤติกรรม ซึ่งโครงสร้างจะเป็นแบบระดับชั้น (Hierarchical) เช่นเดียวกับโครงสร้างของออบเจคคลาส ตัวอย่างเช่น เที่ยวบิน 123 จากชิคาโก และเที่ยวบิน 456 จากโรมคือสมาชิกของอีเวนท์คลาส "เที่ยวบินขึ้น" อีเวนท์คลาสมีแอทริบิวท์ระบุข้อมูลที่นำไปเช่นเดียวกับข้อมูลในออบเจค โดยเขียนแอทริบิวท์ในวงเล็บหลังชื่อของอีเวนท์คลาส ตัวอย่างเช่น

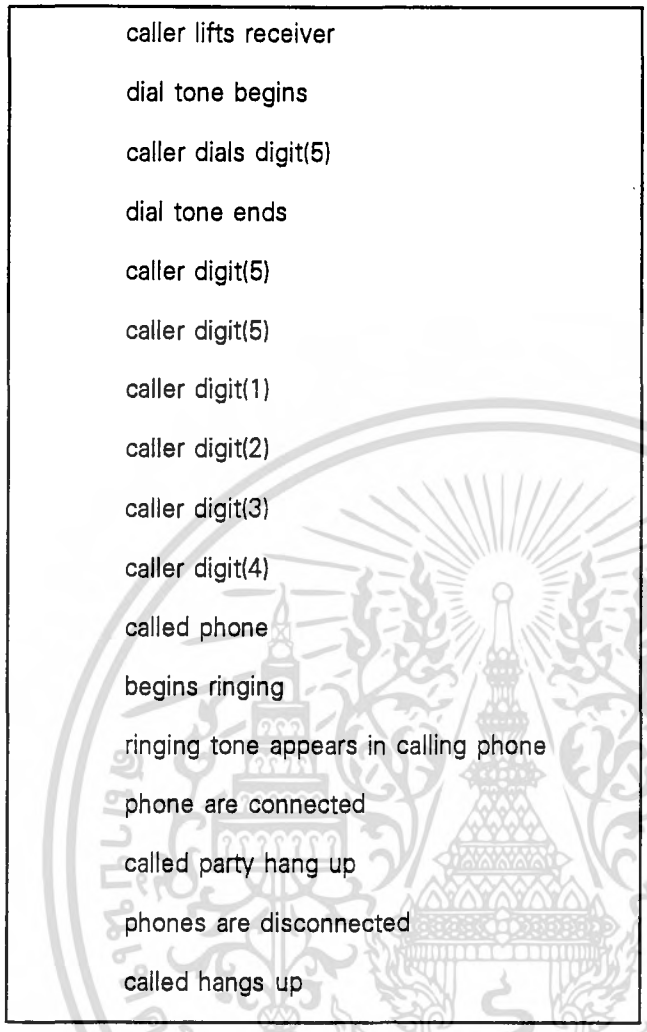
airplane filght departs (airline,filght number,city)

input string entered(text)

phone receiver lifted

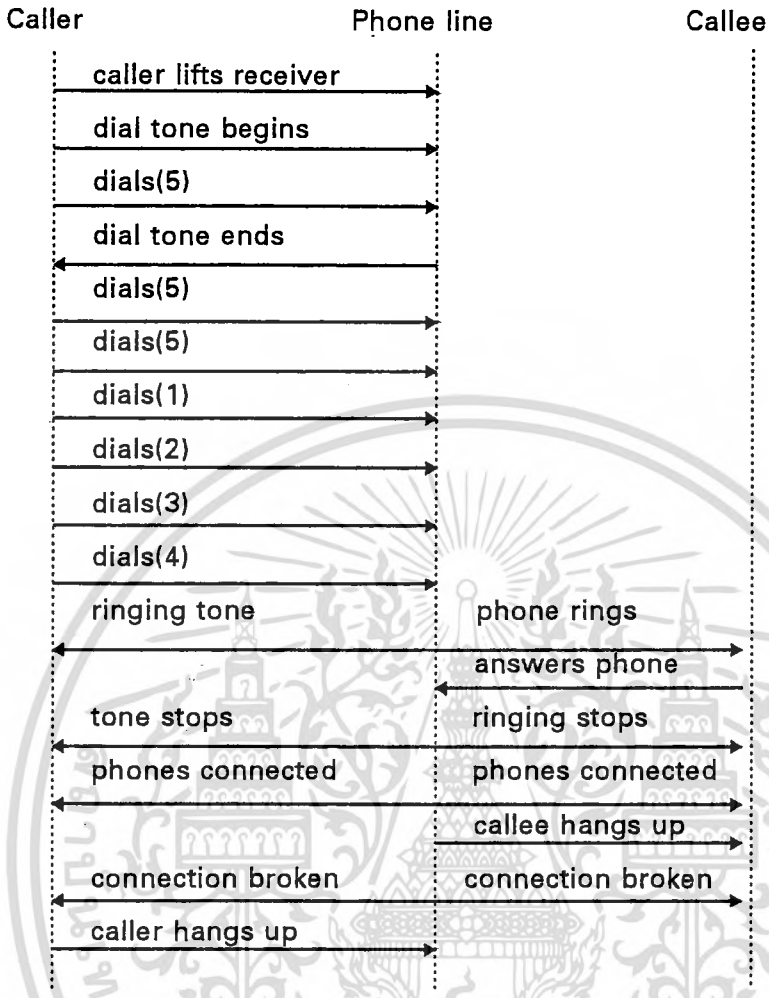
อีเวนท์รวมเงื่อนไขที่ผิดพลาดไว้ด้วยเช่นเดียวกันกับอีเวนท์ที่เกิดขึ้นแบบปกติ ตัวอย่างเช่น การจราจรติดขัด อีเวนท์ที่ผิดพลาดไม่ได้มีอะไรแตกต่างไปจากอีเวนท์ปกติเพียงแต่บอกว่ามันคือข้อผิดพลาด

ภาพเหตุการณ์ (Senarios) คือลำดับของอีเวนท์ซึ่งเกิดขึ้นในเวลาที่ระบบกำลังทำงาน ขอบเขตของภาพเหตุการณ์สามารถเปลี่ยนแปลงได้ ซึ่งอาจจะรวมทุกอีเวนท์ในระบบหรืออาจจะรวมเฉพาะบางออบเจคในระบบ ภาพเหตุการณ์สามารถเป็นรายการในอดีตของการทำงานของระบบ หรือแนวความคิดที่ได้จากประสบการณ์ในการทำงาน ตัวอย่างภาพเหตุการณ์สำหรับการใช้โทรศัพท์ในภาพที่ 18 ซึ่งแสดงเฉพาะอีเวนท์ที่มีผลกระทบกับ phone line เท่านั้น



ภาพที่ 18 ภาพเหตุการณ์สำหรับ phone call

แต่ละอีเวนท์ส่งผ่านข้อมูลจากออบเจคหนึ่งไปยังออบเจคอื่น ตัวอย่างเช่น dial tone begins ส่งสัญญาณจากโทรศัพท์ไปยังผู้เรียก ขั้นตอนต่อไปหลังจากเขียนภาพเหตุการณ์คือการกำหนดออบเจคที่เป็นผู้รับและผู้ส่งของแต่ละอีเวนท์ ลำดับของอีเวนท์และออบเจคสามารถแสดงได้ทั้งสองแบบในภาพเหตุการณ์ที่ขยายเพิ่มขึ้นเรียกว่าแผนผังทางเดินของเหตุการณ์ (Event Trace Diagram) ซึ่งแสดงออบเจคแต่ละออบเจคด้วยเส้นดิ่งและแต่ละอีเวนท์ด้วยลูกศรในแนวนอนจากออบเจคผู้ส่งถึงออบเจคผู้รับ เริ่มจากบรรทัดแรกจนถึงบรรทัดสุดท้ายตามระยะเวลา โดยช่องว่างระหว่างลูกศรไม่มีความหมายอะไรเพียงแต่เป็นลำดับของอีเวนท์เท่านั้น ตัวอย่างทางเดินของเหตุการณ์สำหรับ phone call ดังแสดงในภาพที่ 19



ภาพที่ 19 ทางเดินของเหตุการณ์สำหรับ phone call

2.1.2.2 **สเตท**

สเตทคือนามธรรมของค่าแอทริบิวท์และลิงค์ของออบเจคซึ่งรวมค่าต่างๆ เข้าด้วยกันเป็นสเตท โดยรวมค่าเป็นกลุ่มตามคุณสมบัติที่มีอิทธิพลกับพฤติกรรมส่วนใหญ่ของออบเจค ตัวอย่างเช่น สเตทของธนาคารคือสามารถชำระหนี้ได้หรือไม่สามารถชำระหนี้ได้ซึ่งขึ้นกับทรัพย์สินของธนาคาร สเตทกำหนดการตอบสนองของออบเจคต่ออีเวนท์ที่เข้ามายังออบเจคซึ่งการตอบสนองอาจจะเปลี่ยนแปลงตามค่าของแอทริบิวท์ แต่การตอบสนองจะเป็นแบบเดียวกันสำหรับทุกๆ ค่าของข้อมูลในสเตทเดียวกันและอาจจะแตกต่างกันสำหรับค่าที่อยู่ในสเตทที่ต่างกัน การตอบสนองของออบเจคต่ออีเวนท์อาจจะรวมแอทชันหรือการเปลี่ยนสเตทโดยออบเจค ตัวอย่างเช่น ถ้าตัวเลขถูกหมุนในสเตท Dial tone และออบเจค phone line ออกจากสเตท Dial

tone เข้าสู่สเตท Dialing ถ้าออบเจค receiver เข้ามาแทนที่ ออบเจค phone line จะตายไป และเข้าสู่สเตท Idle สเตทอยู่ตรงกลางระหว่างสองอีเวนท์ซึ่งเป็นอีเวนท์ที่ออบเจคได้รับ อีเวนท์แทนจุดในเวลานั้น สเตทแทนช่วงระยะเวลา ตัวอย่างเช่น หลังจากยกหูฟังและก่อนหมุนตัวเลขตัวแรก phone line จะอยู่ในสเตท Dial tone สเตทของออบเจคขึ้นกับลำดับของอีเวนท์ซึ่งออบเจคได้รับ ในเกือบทุกกรณีอีเวนท์ในอดีตจะถูกซ่อนไว้โดยอีเวนท์ที่ตามมา ตัวอย่างเช่น อีเวนท์ซึ่งเกิดขึ้นก่อนวางหูฟังไม่มีผลกับพฤติกรรมในอนาคต สเตท Idle ลืมอีเวนท์ที่ได้รับก่อนอีเวนท์วางหู สเตทที่มีช่วงระยะเวลามักเกิดขึ้นในช่วงเวลาหนึ่ง สเตทเกี่ยวข้องกับการกระทำที่ต่อเนื่อง เช่น การทำให้เสียงโทรศัพท์ดัง หรือสเตทเกี่ยวข้องกับแอคติวิตี้ (Activity) ที่ต้องใช้เวลาทำให้เสร็จสมบูรณ์ เช่น เที่ยวบินจากชิกาโกถึงซานฟรานซิสโก อีเวนท์และสเตทเป็นสิ่งที่คู่กันอีเวนท์หนึ่งจะแบ่งแยกสองสเตทและสเตทหนึ่งจะแบ่งแยกสองอีเวนท์ สเตทมักเกี่ยวข้องกับค่าของออบเจคซึ่งตรงกับเงื่อนไขใดเงื่อนไขหนึ่ง ค่าของแอทริบิวท์แต่ละค่ากำหนดสเตทที่แตกต่างกัน

เงื่อนไข (Condition) คือบูลลีนฟังค์ชัน (Boolean Function) ของค่าออบเจค เช่น 'อุณหภูมิต่ำกว่าจุดแข็ง' เงื่อนไขจะเป็นจริงตามระยะเวลา เช่น "อุณหภูมิต่ำกว่าจุดแข็งจาก 15 พ.ย 1921 จนกระทั่งถึง 3 มี.ค 1922" การแยกเงื่อนไขจากอีเวนท์ซึ่งไม่มีช่วงของระยะเวลาเป็นเรื่องที่สำคัญ สเตทสามารถกำหนดในรูปของเงื่อนไขได้ ในทางกลับกันสเตทคือเงื่อนไข เงื่อนไขสามารถใช้เป็นเสมือนผู้คุมทรานซิชัน (Transition) ทรานซิชันที่มีผู้คุมนี้จะเกิดขึ้นได้ก็ต่อเมื่อมีอีเวนท์เกิดขึ้น และอีเวนท์เกิดขึ้นได้ก็เพราะเงื่อนไขเป็นจริง เช่น "เมื่อคุณออกจากบ้านในตอนเช้า ถ้าฝนตก (เงื่อนไข) ดังนั้นคุณกางร่ม (สเตทถัดไป)" เงื่อนไขที่คุมทรานซิชันแสดงในรูปของนิพจน์บูลลีน ภายในวงเล็บตามหลังอีเวนท์

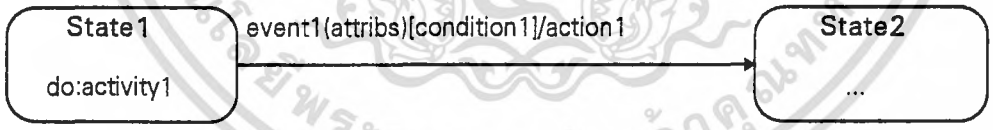
2.1.2.3 โอเปอเรชัน (Operation)

สเตทไดอะแกรมจะมีประโยชน์เพียงเล็กน้อยเท่านั้นถ้าใช้อธิบายเฉพาะรูปแบบของอีเวนท์ การอธิบายพฤติกรรมของออบเจคต้องกำหนดว่าออบเจคตอบสนองต่ออีเวนท์อย่างไร การที่โอเปอเรชันถูกกระทำก็เพื่อตอบสนองต่อสเตทหรืออีเวนท์ แอคติวิตี้คือโอเปอเรชันซึ่งต้องใช้เวลาเพื่อให้การกระทำสมบูรณ์ แอคติวิตี้เกี่ยวข้องกับสเตท แอคติวิตี้รวมโอเปอเรชันทั้งแบบต่อเนื่องและแบบเรียงลำดับ ตัวอย่างโอเปเรชันแบบต่อเนื่องเช่น ภาพบนจอโทรทัศน์ สัญญลักษณ์ของแอคติวิตี้คือการเขียนชื่อของแอคติวิตี้ตามหลัง "do:" ภายในสัญญลักษณ์รูปกล่องสี่เหลี่ยมของสเตท เช่น "do: A" โดยแอคติวิตี้ A จะเริ่มทำงานเมื่อเข้าสู่สเตทและจะสิ้นสุดการทำงานเมื่อออกจากสเตท สเตทอาจจะควบคุมลำดับของแอคติวิตี้ เช่น การเคลื่อนที่ของหุ่นยนต์ซึ่งจะกระทำจน

กว่าจะสมบูรณ์ หรือจนกว่าจะถูกเบรคทำให้ต้องหยุดการกระทำก่อนเวลา แอคชัน(Action) คือ โอเปอเรชันชั่วขณะ แอคชันเกี่ยวข้องกับอีเวนท์ แอคชันแทนโอเปอเรชันซึ่งช่วงระยะเวลาไม่มีความสำคัญในสเตทไดอะแกรม ความจริงแล้วโอเปอเรชันชั่วขณะจริงๆ ไม่มีแต่จำลองโอเปอเรชันเป็นแอคชันนี้เพื่อชี้ว่าเราไม่สนใจช่วงระยะเวลา ถ้าหากว่าเราสนใจช่วงระยะเวลาเราจะจำลองโอเปอเรชันเป็นแอคติวิตี้ซึ่งมีอีเวนท์เริ่มต้น อีเวนท์จบ และอีเวนท์ภายในช่วงระยะเวลา เขียนชื่อของแอคชันตามหลังชื่อของอีเวนท์ โดยหน้าชื่อของแอคชันมีเครื่องหมาย "/"

2.1.2.4 สเตทไดอะแกรม

สเตทไดอะแกรมเกี่ยวข้องกับอีเวนท์และสเตท อีเวนท์ทำให้เกิดการเปลี่ยนสเตท ซึ่งเรียกการเปลี่ยนสเตทว่าทรานซิชัน (Transition) สเตทไดอะแกรมคือกราฟที่มีสเตทเป็นโหนด (Node) และเส้นที่ลากต่อระหว่างโหนดคือทรานซิชัน โดยเขียนชื่อของอีเวนท์ที่สเตทนั้นได้รับกำกับบนทรานซิชัน สเตทใช้สัญญลักษณ์เป็นรูปสี่เหลี่ยมมุมมน และเขียนชื่อของสเตทภายในรูปสี่เหลี่ยมหรือจะไม่เขียนก็ได้โดยเขียนด้วยตัวอักษรทึบ เส้นลูกศรลากออกจากสเตทหนึ่งไปยังสเตทเป้าหมายแทนทรานซิชัน เขียนชื่อของอีเวนท์บนทรานซิชัน ซึ่งอาจจะมีแอทริบิวท์เขียนอยู่ภายในวงเล็บตามหลังหรือไม่มีก็ได้ ถ้ามีเงื่อนไขจะเขียนภายในเครื่องหมายวงเล็บใหญ่ต่อจากชื่อของอีเวนท์ และเขียนชื่อของแอคชันต่อหลังโดยมีเครื่องหมาย "/" นำหน้าชื่อแอคชัน ส่วนชื่อของแอคติวิตี้จะเขียนอยู่ในรูปกล่องสเตทหลังคำสั่ง "do:" ดังแสดงในภาพที่ 20



ภาพที่ 20 สเตทไดอะแกรม

สเตทไดอะแกรมกำหนดลำดับของสเตทซึ่งเป็นผลมาจากลำดับของอีเวนท์ ลำดับของอีเวนท์ตรงกับเส้นทางบนกราฟ สเตทไดอะแกรมอธิบายพฤติกรรมของแต่ละออบเจคคลาส เมื่อทุกสมาชิกของคลาสมีพฤติกรรมแบบเดียวกัน (โดยคำจำกัดความ) สมาชิกจะใช้สเตทไดอะแกรมร่วมกันเช่นเดียวกับที่ใช้ลักษณะของคลาสร่วมกัน เพียงแต่ว่าแต่ละออบเจคจะมีสเตทเป็นของตนเองและลำดับของอีเวนท์ที่ออบเจคได้รับจะแตกต่างกัน แต่ละออบเจคจะเป็นอิสระต่อกันและมีการกระทำในตำแหน่งของตนเอง สเตทไดอะแกรมสามารถแทนวงรอบแบบสั้นๆ (One-shot life

cycle) หรือแบบต่อเนื่องได้ ไดอะแกรมแบบวงรอบสั้นๆ จะมีสเตทเริ่มต้น (Initial) และสเตทสุดท้าย (Final) สเตทเริ่มต้นสร้างออบเจคแต่สเตทสุดท้ายทำลายออบเจค สัญญลักษณ์ของสเตทเริ่มต้นเป็นวงกลมทึบ ซึ่งอาจจะกำหนดชื่อเพื่อระบุเงื่อนไขการเริ่มต้นที่แตกต่างกัน สเตทสุดท้ายใช้สัญญลักษณ์รูปลูกตาวัว (Bull's eye ◉) โดยจะกำหนดชื่อเพื่อให้สามารถที่จะแยกความแตกต่างของเงื่อนไขการจบได้ เราสามารถที่จะมองไดอะแกรมของวงรอบสั้นๆ เป็นซับรูทีนของสเตทไดอะแกรมได้ ซึ่งสามารถจะอ้างถึงได้หลายแห่งในไดอะแกรม ไดนามิกโมเดลคือสเตทไดอะแกรมหลายๆ ไดอะแกรมรวมกัน ออบเจคโมเดลแสดงโครงสร้างของระบบแบบคงที่ ขณะที่ไดนามิกโมเดลแสดงการควบคุมโครงสร้างของระบบ สเตทไดอะแกรมเหมือนกับออบเจคคลาสที่เป็นรูปแบบอธิบายรายละเอียดทั้งหมดและภาพเหตุการณ์เหมือนกับเป็นสมาชิกของไดนามิกโมเดล

2.1.3 ฟังก์ชันนัลโมเดล

ฟังก์ชันนัลโมเดลแสดงผลลัพธ์การคำนวณโดยไม่ได้กำหนดว่าคำนวณอย่างไรและเมื่อไร ฟังก์ชันนัลโมเดลกำหนดความหมายของโอเปอเรชันในออบเจคโมเดล และแอคชันในไดนามิกโมเดล รวมทั้งข้อบังคับในออบเจคโมเดล โปรแกรมแบบไม่มีการโต้ตอบ เช่น คอมไพเลอร์มีไดนามิกโมเดลเพียงเล็กน้อยเพราะจุดมุ่งหมายของโปรแกรมคือฟังก์ชันการคำนวณซึ่งฟังก์ชันนัลโมเดลคือโมเดลหลักสำหรับโปรแกรมเช่นนี้ ออบเจคโมเดลสำคัญสำหรับปัญหาแบบโครงสร้างข้อมูล โปรแกรมแบบที่มีการโต้ตอบหลายโปรแกรมก็มีฟังก์ชันนัลโมเดลที่สำคัญด้วยเช่นกัน ตรงกันข้ามกับฐานข้อมูลมักจะมีฟังก์ชันนัลโมเดลเพียงเล็กน้อยเมื่อจุดประสงค์คือการเก็บและการจัดข้อมูลไม่ใช่การเคลื่อนย้ายข้อมูล ฟังค์ชันนัลโมเดลประกอบด้วยดาต้าโฟลไดอะแกรม (Data Flow Diagram) หลายไดอะแกรมซึ่งกำหนดความหมายของโอเปอเรชันและข้อบังคับ ดาต้าโฟลไดอะแกรมแสดงความสัมพันธ์แห่งการกระทำของค่าที่คำนวณโดยระบบรวมทั้งค่าของอินพุท ค่าของผลลัพธ์ และที่เก็บข้อมูลภายในระบบ ดาต้าโฟลไดอะแกรมคือกราฟที่แสดงดาต้าโฟลจากแหล่งของข้อมูลในออบเจคผ่านทางโพรเซส (Process) โพรเซสจะส่งข้อมูลต่อไปยังจุดหมายปลายทางในออบเจคอื่นๆ ดาต้าโฟลไดอะแกรมไม่ได้แสดงข้อมูลข่าวสารการควบคุมเช่นเวลาที่โพรเซสทำงานหรือการตัดสินใจเลือกดาต้าโฟล การตัดสินใจเลือกนี้อยู่ในไดนามิกโมเดล ดาต้าโฟลไดอะแกรมไม่ได้แสดงการจัดข้อมูลในออบเจค การจัดข้อมูลนี้อยู่ในออบเจคโมเดล ดาต้าโฟลไดอะแกรมประกอบด้วย

- โพรเซส (Process) ซึ่งทำการส่งผ่านข้อมูล

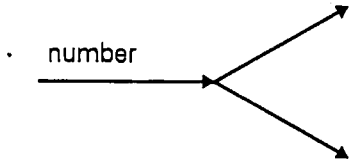
- ดาต้าโฟล (Data Flow) ซึ่งเคลื่อนย้ายข้อมูล
- ผู้กระทำ (actor) ออบเจคซึ่งสร้างและใช้ข้อมูล
- แหล่งเก็บข้อมูล (data store) ออบเจคซึ่งเก็บข้อมูล

2.1.3.1 โพรเซส (Process)

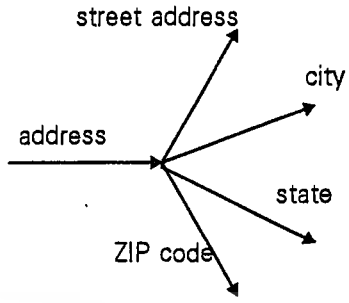
โพรเซสจะส่งผ่านค่าของข้อมูล โพรเซสในระดับต่ำสุดก็คือฟังก์ชันแท้ๆ แผนผังของดาต้าโฟลทั้งหมดคือโพรเซสในระดับสูง ฟังก์ชันนัลโมเดลระบุเฉพาะเส้นทางของการกระทำที่เป็นไปได้เท่านั้นแต่ไม่ได้แสดงว่าเส้นทางใดจะเกิดการกระทำขึ้น ผลลัพธ์ของโพรเซสขึ้นกับพฤติกรรมของระบบตามที่กำหนดไว้ในไดนามิกโมเดล สัญญลักษณ์ของโพรเซสคือวงรีรูปไข่โดยเขียนชื่อของโพรเซสไว้ภายใน แต่ละโพรเซสมีลูกศรที่เป็นข้อมูลอินพุทและเอาว์พุทเป็นจำนวนที่แน่นอน ซึ่งแต่ละลูกศรมีค่าของข้อมูลตามชนิดที่กำหนดให้ เราสามารถที่จะกำหนดชื่อของอินพุทและเอาว์พุทเพื่อระบุบทบาทในการคำนวณได้ โพรเซสสามารถมีผลลัพธ์ได้มากกว่าหนึ่งผลลัพธ์ แผนผังแสดงเฉพาะรูปแบบของอินพุทและเอาว์พุท ต้องกำหนดการคำนวณค่าของผลลัพธ์จากค่าของอินพุทด้วย โพรเซสในระดับสูงสามารถขยายออกเป็นดาต้าโฟลไดอะแกรมได้มากเท่าที่ซับรูทีนสามารถแตกออกเป็นซับรูทีนในระดับต่ำ โพรเซสซึ่งแยกย่อยออกไปอีกไม่ได้แล้วต้องอธิบายด้วยภาษาที่เข้าใจได้ง่ายหรือเป็นสมการการคำนวณ โพรเซสสร้างขึ้นเป็นเมธธอด (หรือส่วนหนึ่งของเมธธอด) ของโอเปอเรชันในออบเจค

2.1.3.2 ดาต้าโฟล (Data Flow)

ดาต้าโฟลเชื่อมต่อระหว่างผลลัพธ์ของออบเจค หรือโพรเซสกับอินพุทจากออบเจคหรือโพรเซสอื่น ดาต้าโฟลแทนค่าของข้อมูลที่เกิดจากการคำนวณ ค่าของข้อมูลไม่ได้เปลี่ยนแปลงไปเพราะดาต้าโฟล ดาต้าโฟลแทนด้วยเส้นลูกศรซึ่งอยู่ระหว่างผู้สร้างและผู้ใช้ค่าของข้อมูล โดยบนลูกศรจะมีคำอธิบายเกี่ยวกับค่าของข้อมูล ซึ่งปกติจะเขียนเป็นชื่อหรือเป็นชนิดของข้อมูล ค่าเดียวกันสามารถส่งไปได้หลายที่ ซึ่งจะใช้สัญญลักษณ์เป็นซ่อมที่มีหลายหัวลูกศรรวมกัน โดยที่ลูกศรผลลัพธ์ไม่ได้เขียนชื่อกำกับเพราะว่าใช้แทนค่าอินพุทอันเดียวกัน ดังแสดงตัวอย่างในภาพที่ 21 ในบางครั้งค่าของข้อมูลแบบอะกรีเกชันแตกออกเป็นส่วนประกอบซึ่งแต่ละส่วนประกอบถูกส่งไปยังโพรเซสที่แตกต่างกัน ในกรณีนี้จะแสดงด้วยสัญญลักษณ์ซ่อมซึ่งแต่ละลูกศรจะเขียนชื่อของส่วนประกอบกำกับ ส่วนประกอบที่รวมกันเป็นค่าอะกรีเกชันคือค่าที่อยู่ด้านตรงข้าม ดังแสดงตัวอย่างในภาพที่ 22



ภาพที่ 21 ตัวอย่างดาต้าโฟล



ภาพที่ 22 ตัวอย่างดาต้าโฟล

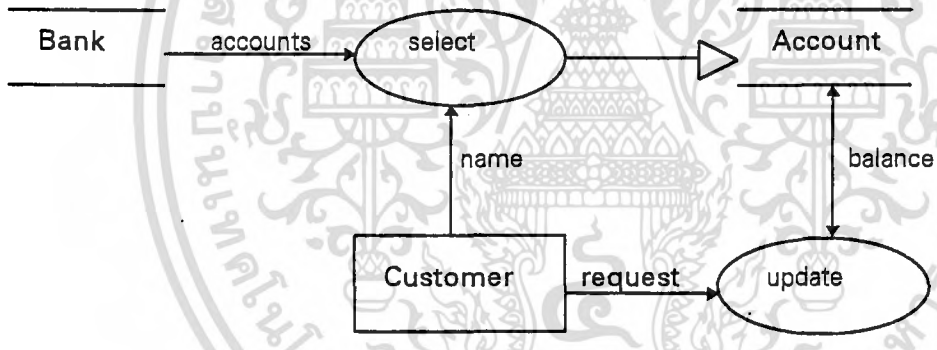
แต่ละดาต้าโฟลจะแทนค่าซึ่งได้จากการคำนวณที่จุดใดจุดหนึ่ง ซึ่งในความเป็นจริงค่านี้ไม่จำเป็นต้องมีความสำคัญ โฟลที่อยู่ตรงส่วนขอบของดาต้าโฟลไดอะแกรมคืออินพุทและเอาว์พุท โฟลเหล่านี้อาจจะไม่ถูกเชื่อมต่อ (ถ้าไดอะแกรมเป็นเพียงส่วนหนึ่งของระบบ) หรืออาจจะเชื่อมต่อกับออบเจค

ผู้กระทำ (Actor) คือออบเจคที่มีการกระทำที่ขับเคลื่อนดาต้าโฟลโดยการสร้างและการใช้ค่า ผู้กระทำติดต่อกับอินพุทและเอาว์พุทของดาต้าโฟลผู้กระทำวางอยู่บนขอบนอกของกราฟดาต้าโฟลแต่จบการไหลของข้อมูลด้วยแหล่งข้อมูลและเก็บข้อมูลไว้ ดังนั้นในบางครั้งจึงเรียกผู้กระทำนี้ว่าเทอร์มิเนเตอร์ (Terminators) การกระทำของผู้กระทำอยู่นอกขอบเขตของดาต้าโฟลไดอะแกรมแต่ควรจะเป็นส่วนหนึ่งของไดนามิกโมเดล ผู้กระทำแทนด้วยรูปสามเหลี่ยมเพื่อแสดงว่าเป็นออบเจค ลูกศรระหว่างผู้กระทำและไดอะแกรมคืออินพุทและเอาว์พุทของไดอะแกรม

แหล่งเก็บข้อมูล (Data Store) คือออบเจคที่ไม่มีการกระทำภายในดาต้าโฟลไดอะแกรมแต่เก็บข้อมูลสำหรับการกระทำล่าสุด แหล่งเก็บข้อมูลไม่เหมือนกับผู้กระทำเพราะแหล่งเก็บข้อมูลไม่ได้สร้างโอเปอเรชันใดๆ บนแหล่งเก็บข้อมูลเพียงแต่ตอบรับที่จะเก็บข้อมูล แหล่งเก็บข้อมูลเขียนแทนด้วยเส้นคู่ขนานที่เขียนชื่อแหล่งเก็บข้อมูลไว้ภายใน ลูกศรเข้าไปที่แหล่งเก็บข้อมูลระบุข้อมูลข่าวสารหรือโอเปอเรชันซึ่งปรับปรุงแก้ไขข้อมูลที่เก็บไว้ในแหล่งเก็บข้อมูล ได้แก่ การเพิ่ม การเปลี่ยนแปลงค่า หรือการลบค่าออก ลูกศรออกจากแหล่งเก็บข้อมูลระบุข้อมูลข่าวสารที่ถูกดึงออกมาจากแหล่งเก็บข้อมูล ซึ่งอาจจะเป็นค่าทั้งหมดหรือค่าเพียงบางส่วน โครงสร้างของออบเจคจะต้องอธิบายในออบเจคโมเดล รวมทั้งต้องอธิบายโอเปอเรชันการแก้ไขและการกระทำ

ทั้งผู้กระทำและแหล่งเก็บข้อมูลคือออบเจค แต่ที่เรากำหนดให้แตกต่างกันก็เพราะพฤติกรรมและการใช้มีความแตกต่างกันถึงแม้ว่าในภาษาเชิงวัตถุจะสร้างเป็นออบเจคเหมือนกัน หรือแหล่งเก็บข้อมูลอาจจะสร้างเป็นแฟ้มข้อมูลและผู้กระทำเป็นอุปกรณ์ภายนอก ดาต้าโฟลดาต้าโฟลเป็นออบเจคด้วย ถึงแม้ว่าในหลายกรณีที่ดาต้าโฟลคือค่าแท้ๆ เช่น เลขจำนวนเต็ม (ในภาษาเชิงวัตถุออบเจคและค่าแท้ๆ มักจะสร้างเหมือนกัน)

มุมมองที่แตกต่างกันระหว่างออบเจคคือออบเจคมีข้อมูลได้ค่าเดียวแต่แหล่งเก็บข้อมูลมีข้อมูลได้มากกว่าหนึ่งค่า ตัวอย่างเช่นในภาพที่22 นำชื่อของลูกค้าไปเลือกบัญชีจากธนาคารได้ผลลัพธ์เป็นออบเจคบัญชีนั้นซึ่งใช้เป็นแหล่งเก็บข้อมูลสำหรับโอเปอเรชันการแก้ไข ดาต้าโฟลซึ่งสร้างออบเจคและจะใช้เป็นเป้าหมายของโอเปอเรชันอื่นระบุโดยใช้รูปสามเหลี่ยมกลวงที่ปลายของดาต้าโฟล ตรงกันข้ามโอเปอเรชันการแก้ไขซึ่งปรับยอดบัญชีในออบเจคบัญชีระบุโดยใช้หัวลูกศรเล็กๆ รูปสามเหลี่ยมกลวงที่ระบุค่าผลลัพธ์ของดาต้าโฟลเป็นออบเจคโดยปกติแล้วก็คือแหล่งเก็บข้อมูล



ภาพที่ 23 ตัวอย่างดาต้าโฟลไดอะแกรม

การควบคุมโฟลของข้อมูล (Control Flow) ดาต้าโฟลไดอะแกรมแสดงเส้นทางการคำนวณที่เป็นไปได้สำหรับค่าข้อมูล โดยไม่แสดงว่าเส้นทางใดถูกกระทำและลำดับของการกระทำเป็นอย่างไร การตัดสินใจและการเรียงลำดับคือการควบคุมซึ่งเป็นส่วนหนึ่งของไดนามิกโมเดล การตัดสินใจนั้นมีผลกับฟังค์ชันว่ามีหนึ่งฟังค์ชันหรือมากกว่าหนึ่งฟังค์ชันที่จะถูกกระทำไม่ใช่เป็นการให้ค่ากับฟังก์ชัน ถึงแม้ว่าการตัดสินใจเลือกฟังก์ชันนี้จะไม่ได้มีค่าอินพุทแต่ในบางครั้งก็มีประโยชน์ที่จะรวมการตัดสินใจเลือกฟังก์ชันนี้ไว้ในฟังค์ชันนัลโมเดลเพื่อว่าจะได้ไม่ลืมและยังเป็นการแสดงข้อมูลที่ขึ้นต่อกันอีกด้วยวิธีทำคือรวมการควบคุมดาต้าโฟลนี้เข้าไปไว้ในดาต้าโฟลไดอะแกรมด้วย การควบคุมดาต้าโฟลคือค่าบลูลีน (Boolean) ซึ่งจะมีผลกับโพรเซสว่าว่าโพร

เซสใดจะถูกกระทำ ตัวของการควบคุมดาต้าโฟลไม่ใช่เป็นค่าอินพุทของโพรเซส การควบคุมดาต้าโฟลแทนด้วยเส้นลูกศรประจากโพรเซสที่สร้างค่าบลูลีนไปยังโพรเซสที่ถูกควบคุม การควบคุมดาต้าโฟลนั้นมีประโยชน์ แต่จะซ้ำกับข้อมูลข่าวสารในไดนามิกโมเดล ดังนั้นจึงควรใช้เท่าที่จำเป็นเท่านั้น

2.1.3.3 การกำหนดโอเปอเรชัน

โพรเซสในดาต้าโฟลไดอะแกรมสร้างเป็นโอเปอเรชันในออบเจค แต่ละโอเปอเรชันอาจจะกำหนดได้หลายวิธีดังต่อไปนี้

-ฟังก์ชันทางคณิตศาสตร์

-ตารางของค่าอินพุทและเอาว์พุท (ตัวเลข) สำหรับกลุ่มข้อมูลเล็กๆ

-สมการการกำหนดผลลัพธ์ในเทอมของอินพุท

-เงื่อนไขก่อนและหลัง (คำจำกัดความ)

-ซูโดโคด(pseudocode)

-ภาษาธรรมชาติ

โดยปกติแล้วโอเปอเรชันจะเขียนอยู่ในออบเจคโมเดล เพื่อแสดงรูปแบบของการถ่ายทอดโอเปอเรชันซึ่งอ่านหรือเขียนแอทริบิวท์หรือลิงค์ของออบเจค ในระหว่างการวิเคราะห์ยังไม่จำเป็นต้องกำหนดโอเปอเรชันนี้ ในช่วงของการออกแบบจึงจำเป็นที่จะต้องกำหนดโดยกำหนดว่าโอเปอเรชันใดจะเป็นแบบสาธารณะและแบบส่วนบุคคลสำหรับออบเจคคลาส ซึ่งเหตุผลการแบ่งประเภทของโอเปอเรชันนี้เพื่อเป็นการหุ้มห่อออบเจคคลาส (Encapsulation) ไว้เป็นการป้องกันความผิดพลาด เปอเรชันสามารถแบ่งออกได้เป็น 3 กลุ่มด้วยกันคือ

- คิวรี่ (Query)
- แอคชัน (Action)
- แอคติวิตี้ (Activity)

คิวรี่คือฟังก์ชันแท้ๆ ซึ่งไม่มีผลกระทบกับสเตทที่มองเห็นได้จากภายนอกออบเจค คิวรี่ที่ไม่มีพารามิเตอร์ (ยกเว้นสำหรับออบเจคเป้าหมาย) คือดิไรว์แอทริบิวท์ (derived attribute) ดังนั้นจะมีรูปแบบของแอทริบิวท์ ตัวอย่างเช่น จุดที่กำหนดในคาร์ทีเซียนโคออดิเนต (Cartesian Coordinate) นั่นคือรัศมีและมุมคือดิไรว์แอทริบิวท์ ในออบเจคโมเดลคิวรี่โอเปอเรชันสามารถรวมเข้ากับแอทริบิวท์ได้เพราะว่าคิวรี่โอเปอเรชันไม่มีผลกระทบกับภายนอก และมีความสำคัญน้อยกว่าแอทริบิวท์และแอคชันในช่วงเวลาของการวิเคราะห์และออกแบบระบบ

แอคชันคือการแปลงค่าซึ่งมีผลกระทบกับออบเจคเป้าหมายหรือออบเจคเป้าหมายอื่นๆ ในระบบซึ่งเข้าถึงได้ แอคชันไม่มีช่วงระยะเวลามันจะเกิดขึ้นทันทีทันใด (ถึงแม้ว่าตามปกติแล้วจะต้องใช้เวลาบางก็ตาม) เพราะว่าสเตทของออบเจคกำหนดโดยแอทริบิวท์และลิงค์ ทุกแอคชันต้องถูกกำหนดในเทอมของการแก้ไขแอทริบิวท์และลิงค์ได้ ปกติแอคชันได้มาจากโพรเซสในฟังค์ชันนัลโมเดล แอคชันสามารถอธิบายได้หลายทางรวมทั้งสมการทางคณิตศาสตร์ ขบวนการตัดสินใจแบบทรี (Decision Tree) หรือแบบตาราง (Decision Table) ตัวเลขของทุกๆ อินพุทที่เป็นไปได้ แคลคูลัส และภาษาธรรมชาติ สิ่งที่สำคัญคือการกำหนดต้องชัดเจนและไม่คลุมเครือ วิธีกำหนดแอคชันวิธีหนึ่งคือการใช้อะกอริทึมสำหรับการคำนวณ บ่อยครั้งที่ฟังค์ชันกำหนดได้ง่ายแต่อะกอริทึมไม่ง่าย ดังนั้นไม่ได้หมายความว่าโปรแกรมจะต้องใช้อะกอริทึมแบบเดียวกันถึงแม้ว่าผลลัพธ์ที่ได้จะเหมือนกันก็ตาม

แอคติวิตี้คือโอเปอเรชันที่ต้องใช้ระยะเวลาตรงกันข้ามกับคิวรี่และแอคชัน แอคติวิตี้มีผลกระทบอันเนื่องมาจากระยะเวลาที่มันใช้

2.1.3.4 ข้อบังคับ (Constraint)

ข้อบังคับแสดงความสัมพันธ์ระหว่างออบเจคสองออบเจคในเวลาเดียวกัน (เช่น ความถี่และความกว้างของคลื่น) หรือความสัมพันธ์ระหว่างค่าที่แตกต่างกันของออบเจคเดียวกันในเวลาที่แตกต่างกัน (เช่น จำนวนหุ้นที่เด่นๆ ของกองทุนเดียวกัน) ข้อบังคับอาจจะแสดงในรูปของฟังค์ชันรวม (ค่าๆ หนึ่งถูกกำหนดอย่างสมบูรณ์โดยค่าอื่นๆ) หรือบางส่วนของฟังค์ชัน (ค่าๆ หนึ่งถูกจำกัด แต่ไม่ถูกกำหนดอย่างสมบูรณ์โดยค่าอื่นๆ) ตัวอย่างเช่นการแปลงค่าโคออดิเนต (Co-ordinate) ต้องกำหนดเฟกเตอร์ของสเกลสำหรับเอกซ์โคออดิเนต(X-Coordinate) และวายโคออดิเนต (Y-Coordinate) ข้อบังคับนี้จะกำหนดค่าๆ หนึ่งในเทอมของอีกค่าหนึ่ง ข้อบังคับสามารถปรากฏได้ในโมเดลแต่ละแบบ ข้อบังคับของออบเจคกำหนดให้ออบเจคบางออบเจคขึ้นกับออบเจคอื่นทั้งหมดหรือขึ้นกับออบเจคอื่นเพียงบางส่วน ข้อบังคับของไดนามิกกำหนดความสัมพันธ์ระหว่างสเตทหรืออีเวนท์ที่ต่างกัน ข้อบังคับของฟังก์ชันนัลกำหนดข้อจำกัดบนโอเปอเรชัน ข้อบังคับระหว่างค่าของออบเจคในเวลามักจะเรียกว่าอินแวเรียนท์ Invariant) ซึ่งมีประโยชน์ในการกำหนดพฤติกรรมของโอเปอเรชัน

2.1.3.5 ความสัมพันธ์ของฟังก์ชันนัลกับออบเจคและไดนามิกโมเดล

ฟังก์ชันนัลโมเดลแสดงว่าอะไรถูกกระทำโดยระบบ โพรเซสที่ไม่สามารถแยกเป็นส่วนที่

ย่อยลงไปได้อีก (Leaf Process) ก็คือโอเปอเรชันในออบเจค ออบเจคโมเดลแสดงผู้กระทำซึ่งก็คือออบเจคนั่นเอง แต่ละโพรเซสถูกสร้างขึ้นมาโดยเมธธอด ไดนามิกโมเดลแสดงลำดับของโอเปอเรชันที่ถูกกระทำ เมธธอดได้มาจากทั้งสามโมเดลรวมกันแต่ฟังก์ชันนัลโมเดลแนะแนวทางเมธธอด โพรเซสในฟังก์ชันนัลโมเดลตรงกับโอเปอเรชันในออบเจคโมเดล การจับคู่กันในเนตเวิร์คโพรเซสระดับบนสุดจะตรงกับโอเปอเรชันของออบเจคที่ซับซ้อนและโพรเซสระดับต่ำสุดจะตรงกับออบเจคขั้นพื้นฐานซึ่งเป็นส่วนหนึ่งของออบเจคที่ซับซ้อน บางครั้งโพรเซสเดียวอาจจะตรงกับหลายโอเปอเรชัน และบางครั้งโอเปอเรชันเดียวจะตรงกับหลายโพรเซส โพรเซสในฟังก์ชันนัลโมเดลแสดงออบเจคซึ่งเกี่ยวข้องกันโดยฟังก์ชัน บ่อยครั้งที่อินพุทหนึ่งของโพรเซสสามารถถูกกำหนดเป็นออบเจคเป้าหมาย และอินพุทส่วนที่เหลือกำหนดเป็นพารามิเตอร์ของโอเปอเรชัน ออบเจคเป้าหมายคือไคลอินท์ (Client) ของออบเจคอื่นซึ่งเรียกว่าซับไพลเออร์ (Supplier) เพราะออบเจคเป้าหมายใช้ออบเจคอื่นทำโอเปอเรชันไคลอินท์ถูกสร้างในเทอมของซับไพลเออร์และขึ้นกับซับไพลเออร์ โดยปกติโพรเซสจะพัฒนาไปเป็นเมธธอด ถ้าอินพุทและเอาว์พุทเป็นออบเจคในคลาสเดียวกันดังนั้นออบเจคคือเป้าหมายและอินพุทที่เหลือคืออาร์กิวเมนต์ ถ้าผลลัพธ์ของโพรเซสคือแหล่งเก็บข้อมูล แหล่งเก็บข้อมูลคือเป้าหมาย ถ้าอินพุทหรือผลลัพธ์คือผู้กระทำ ผู้กระทำคือออบเจคที่กำหนดขึ้นในออบเจคโมเดล ดาต้าโฟลเข้าไปยังผู้กระทำหรือออกจากผู้กระทำใช้แทนโอเปอเรชัน ค่าของดาต้าโฟลคืออาร์กิวเมนต์ของโอเปอเรชันหรือเป็นผลลัพธ์ของโอเปอเรชัน ผู้กระทำคือออบเจคที่มีการเคลื่อนไหวด้วยตนเอง ในฟังก์ชันนัลโมเดลไม่ได้ระบุเมื่อผู้กระทำกระทำแต่กำหนดในไดนามิกโมเดล แหล่งเก็บข้อมูลก็คือออบเจคในออบเจคโมเดลเช่นกัน หรือเป็นส่วนหนึ่งของออบเจค เช่น แอทริบิวท์ ดาต้าโฟลที่เข้าไปยังแหล่งเก็บข้อมูลคือโอเปอเรชันการแก้ไข ดาต้าโฟลที่ออกจากแหล่งเก็บข้อมูลคือโอเปอเรชันคิวรี่ซึ่งไม่มีผลกระทบบนแหล่งเก็บข้อมูล ดาต้าโฟลคือค่าในออบเจคโมเดล ดาต้าโฟลหลายดาต้าโฟลคือค่าแท้ๆแบบธรรมดา เช่น จำนวน ข้อความ เป็นต้น ค่าแท้ๆสามารถถูกจำลองเป็นคลาสได้ และพัฒนาเป็นออบเจคในภาษาส่วนมากแต่ไม่เป็นเอกลักษณ์ (Identity) ค่าแท้ๆไม่ใช่เป็นภาชนะใส่ซึ่งจะเก็บค่าที่สามารถเปลี่ยนแปลงได้แต่เป็นเพียงค่าเท่านั้น เพราะฉะนั้นค่าแท้ๆไม่มีสเตทและไม่มีไดนามิกโมเดล โอเปอเรชันบนค่าแท้ๆ ให้ผลลัพธ์เป็นค่าแท้ๆ และไม่มีผลกระทบ ตัวอย่างเช่นโอเปอเรชันการคำนวณทางคณิตศาสตร์

ฟังก์ชันนัลโมเดล: ออบเจคโมเดลแสดงโครงสร้างของผู้กระทำ, แหล่งเก็บข้อมูล และดาต้าโฟลในฟังก์ชันนัลโมเดล ส่วนไดนามิกโมเดลแสดงลำดับของการทำโพรเซส

ออบเจคโมเดล: ฟังก์ชันนัลโมเดลแสดงโอเปอเรชันบนคลาสและอาร์กิวเมนต์ของแต่ละโอเปอเรชัน ดังนั้นฟังก์ชันนัลโมเดลแสดงความสัมพันธ์ของไคลอึนท์และซับไพลเออร์ในกลุ่มคลาส ไดนามิกโมเดลแสดงสเตทของแต่ละออบเจคและโอเปอเรชันซึ่งกระทำเมื่อได้รับอีเวนท์และเปลี่ยนสเตท

ไดนามิกโมเดล: ฟังก์ชันนัลโมเดลแสดงคำจำกัดความของแอคชันและแอคติวิตี้ที่แยกย่อยลงไปอีกไม่ได้ซึ่งไม่ได้กำหนดไว้ในไดนามิกโมเดล ออบเจคแสดงว่าอะไรเปลี่ยนสเตทและอะไรได้รับโอเปอเรชัน

2.2 โปรแกรมภาษาสมอลทอค (Smalltalk)

โปรแกรมภาษาสมอลทอคนั้นเป็นโปรแกรมเชิงวัตถุอย่างแท้จริง (pure Object-Oriented Programming)

2.2.1 ชนิดของตัวแปร

แบ่งออกเป็น

1) ตัวแปรแบบชั่วคราว (Tempolary Variable)

ตัวแปรแบบชั่วคราวจะกำหนดอยู่ในภายในเครื่องหมาย "|" ที่บรรทัดแรกของชุดคำสั่ง ชื่อของตัวแปรแบบชั่วคราวต้องขึ้นต้นด้วยอักษรตัวเล็ก ส่วนตัวถัดไปจะเป็นตัวอักษรเล็กหรือตัวใหญ่ก็ได้

2) ตัวแปรแบบโกลบอล (Global Variable)

ชื่อของตัวแปรแบบโกลบอลต้องขึ้นต้นด้วยตัวพิมพ์ใหญ่ ส่วนตัวถัดไปจะเป็นตัวเล็ก,ตัวใหญ่ หรือตัวเลขก็ได้

2.2.2 คำสั่ง

- การกำหนดค่า โดย

1) ใช้เครื่องหมาย := (Assignment Operator) กำหนดค่าให้กับตัวแปร

2) ใช้เครื่องหมาย . (Period) แบ่งคำสั่ง(statement) แต่ละคำสั่ง ยกเว้นคำสั่งสุดท้ายในชุดคำสั่ง ตัวอย่างเช่น

```
| temp index factorial |.
factorial := #(3 4 5 6).
```

```
index := 1.
temp := factorial at:index.
index := index + 1
```

สองบรรทัดแรกกำหนดออบเจคให้กับตัวแปรแบบชั่วคราว สองบรรทัดต่อมากำหนดผลลัพธ์ของเมสเซจ (Message) ซึ่งผลลัพธ์ของเมสเซจมักจะเป็นออบเจคเดี่ยวเสมอ โดยปกติมักจะกำหนดออบเจคให้กับตัวแปรแบบชั่วคราว

- การเรียกใช้ฟังก์ชัน

1) แบบมีอาร์กิวเมนต์เดียว

การเรียกใช้ฟังก์ชัน size ในภาษาปาสคาล (Pascal) จะเขียนเป็น

```
a := size(array);
```

แต่การเรียกใช้ฟังก์ชันในภาษาสมอลทอคคือการส่งเมสเซจ ตัวอย่างเช่น

```
a := array size
```

คือการส่งเมสเซจ size ไปยังค่าของตัวแปร array ซึ่งเป็นอาร์กิวเมนต์

2) แบบสองอาร์กิวเมนต์

ในสมอลทอคอาร์กิวเมนต์จะนำหน้าและตามหลังเมสเซจ ตัวอย่างเช่น

Smalltalk	Pascal
x := x1 max:x2.	x := max(x1,x2);
y := p + q.	y := p + q;

เมสเซจ max: ถูกส่งไปยังค่าของ x1 (อาร์กิวเมนต์แรก) กับค่าของ x2 (อาร์กิวเมนต์ที่สอง) แล้วกำหนดผลลัพธ์ที่ได้กับ x

3) แบบอาร์กิวเมนต์ตั้งแต่สามขึ้นไป

ตัวอย่างเช่น

```
a := x between:x1 and:x2
```

เปรียบเทียบกับ Pacal จะเขียนเป็น

```
a := between(x,x1,x2);
```

ในสมอลทอคชื่อของเมสเซจจะ แตกออกเป็นส่วนๆ และชื่อของเมสเซจแต่ละส่วนจะนำหน้าแต่ละอาร์กิวเมนต์ที่ต่อมาจากอาร์กิวเมนต์แรก ในตัวอย่างชื่อเมสเซจคือ between:and:

และอาร์กิวเมนต์คือ x, x1, และ x2

- **การใช้ค่าซับสคริบท์ (Subscript)** ในสมอลทอคใช้เมสเซจ at: และ at:put: ตัวอย่างเช่น

Smalltalk	Pascal
x := a at:i.	x := a[i];
a at:i put:y.	a[i] := y;
a at:i+1 put:(a at:i)	a[i+1] := a[i]

จากตัวอย่างในคำสั่งบรรทัดแรกค่าของ i เป็น index ของอาร์เรย์เพื่อกำหนดค่าของ a และส่งผลลัพธ์ไปเก็บที่ x คำสั่งในบรรทัดที่สองคือแทนค่าของ a ที่ตำแหน่ง i ด้วยค่าของ y คำสั่งในบรรทัดที่สามเป็นการแทนค่าในทำนองเดียวกันเครื่องหมายวงเล็บในสมอลทอคเป็นการกำหนดลำดับของการกระทำ และในสมอลทอคด้านซ้ายมือของเครื่องหมาย := ต้องเป็นตัวแปรแบบสเกลลาร์ (Scalar) เท่านั้น

การเปรียบเทียบออบเจค ในสมอลทอคเปรียบเทียบออบเจคโดยการส่งเมสเซจ <, <=, =, >, >= และ ~= ซึ่งพัฒนาเป็นไบนารีเมสเซจ ดังนั้นเครื่องหมายวงเล็บมักจะต้องใช้ถ้ามีไบนารีเมสเซจอื่น เช่น

```
3<4
#(1 2 3 4) = #(1 2 3 4)
5=(2+3)
```

การเปรียบเทียบมักจะส่งค่ากลับเป็นจริงหรือเท็จ

การทดสอบออบเจค ออบเจคหลายออบเจคเข้าใจเมสเซจซึ่งในบางครั้งทดสอบสเตทของออบเจคหรือเงื่อนไข เช่น

```
$a isUpperCase
('Hello' at:1) isVowel
```

- คำสั่งเงื่อนไข

ภาษาส่วนมากใช้คำสั่ง if สำหรับประโยคเงื่อนไข สมอลทอคใช้บล็อกของโคด (Code) และเมสเซจสำหรับประโยคเงื่อนไข เช่น

```
| max a b |
a := 5 squared.
```

```
b := 4 factorial.
a < b
    ifTrue:[max := b]
    ifFalse:[max := a].
^max
```

เมสเซจการเปรียบเทียบ a < b ส่งค่ากลับเป็นจริงหรือเท็จซึ่งเป็นรีซีพเวอร์ของเมสเซจ ifTrue:ifFalse: การเปรียบเทียบมักจะไม่ใช่แบบเดี่ยวแต่มักจะเป็นแบบผสม (Compound) โดยใช้เมสเซจ and: และ or: ตัวอย่างเช่น (c < $0 or:[c > $9])

รีซีพเวอร์ของเมสเซจ or: คือผลลัพธ์ของการเปรียบเทียบแรกซึ่งเป็นจริงหรือเท็จ ส่วนอาร์กิวเมนต์คือบล็อกของโคดซึ่งเป็นการเปรียบเทียบสุดท้ายมีผลลัพธ์เป็นจริงหรือเท็จเช่นกัน

- **คำสั่งทำซ้ำ (Iterative Statement)**

คำสั่งวนลูปแบบธรรมดาของสมอลทอคใช้เมสเซจ timeRepeat: กำหนดจำนวนครั้งในการทำซ้ำเช่น

```
| string index c |
string:='Now is the time'.
index:=1.
string size timesRepeat:[
        c:=string at:index.
        string at:index put:
            (c isVowel
                ifTrue:[c UpperCase]
                ifFalse:[c asLowerCase]).
        index:=index + 1].
^string
```

เมสเซจวนลูปธรรมดาแบบอื่น เช่น

```
"copy a disk file"
| input output |
input:=File pathName:'go'.
output:=File pathName:'junk'.
[input atEnd]
    whileFalse:[output nextPut:input next].
input close.
output close
```

ในภาษาสมอลทอคข้อความในเครื่องหมายคำพูดคือคำอธิบาย (Comment) ภายในบล็อก input atEnd จะส่งค่ากลับเป็นจริงหรือเท็จ โดยไฟล์ input จะถูกอ่านค่าถัดไปด้วยเมสเซจ next ไฟล์ output จะเขียนค่าอาร์กิวเมนต์ (คือค่าที่ได้จากการอ่านค่าในไฟล์ input) ด้วยเมสเซจ nextPut:

ตัวอย่างของ whileTrue: เช่น

```
...
[ i < 10 ]
whileTrue:[sum:=sum + (a at: i).
i:=i+1]
```

เมสเซจทำซ้ำ to:do: มีอาร์กิวเมนต์ 2 อาร์กิวเมนต์ ตัวอย่างเช่น

```
"draw several polygons"
3 to: 6 do:[:side| side timeRepeat:[Turtle go:6;  turn:360//side]]
```

จากตัวอย่างข้างบน 3 คือรีซีพเวอร์ออบเจค 6 คืออาร์กิวเมนต์ตัวแรกซึ่งเป็นค่าสูงสุดของการทำซ้ำ และ 2 คือบล็อกซองโคดซึ่งมี side เป็นบล็อกอาร์กิวเมนต์ (จะได้กล่าวถึงรายละเอียดในหัวข้อถัดไป) จะเพิ่มค่าทีละ 1 ในแต่ละรอบของการทำซ้ำ แต่ถ้าต้องการจะกำหนดค่าที่จะเพิ่มขึ้นในแต่ละรอบของการทำซ้ำเป็นค่าอื่น สามารถใช้เมสเซจ to:by:do: ได้ตัวอย่างเช่น

```
| sum |
sum:=0.
1/2 to: 1 by 1/8 do:[:i| sum:=sum + 1].
^sum
```

2.2.3 บล็อกอาร์กิวเมนต์

บล็อกอาร์กิวเมนต์คือตัวแปรชั่วคราวแบบหนึ่ง แต่ไม่ต้องกำหนดไว้ที่บรรทัดแรกของชุดคำสั่ง บล็อกอาร์กิวเมนต์เป็นตัวแปรที่ใช้ภายในบล็อกโดยกำหนดเป็นส่วนแรกของบล็อก ซึ่งหน้าตัวแปรมีเครื่องหมาย : (colon) และตัวแปรเขียนแยกกับคำสั่งภายในบล็อกด้วยเครื่องหมาย | (vertical bar)

บล็อกและอาร์กิวเมนต์ทำให้สมอลทอคมีเมสเซจการทำซ้ำหลายเมสเซจ เช่นเมสเซจ do:, select:, reject: และ collect:

- เมสเซจ do:

ตัวอย่างเช่น

```
| vowel |
vowel:=0.
'Now is the time' do:[:char| char isVowel
                              isTrue:[vowel:=vowel + 1]].
^vowel
```

เมสเซจการทำซ้ำ do: มีผลทำให้สตริงค์ส่งตัวอักษรทีละตัวเข้าไปยังบล็อก

- เมสเซจ select:

ตัวอย่างเช่น ('Now is the time' select:[:clc isVowel]) size

เมสเซจการทำซ้ำ select: มีผลทำให้รีซีพเวอร์ส่งค่ากลับเป็นสมาชิกทั้งหมดเข้าไปยังบล็อก ผลลัพธ์คืออักษรสระทั้งหมด และเมสเซจ size จะให้ค่าจำนวนสมาชิกทั้งหมดที่ถูกเลือก

- เมสเซจ reject:

ตัวอย่างเช่น #(1 2 3 4 5 6 7 8 9) reject:[:il i factorial >=(i*i*i)]

การทำงานของเมสเซจ reject: จะทำนองเดียวกับ select: แต่จะส่งค่ากลับเฉพาะสมาชิกทั้งหมดของรีซีพเวอร์ที่ได้ผลลัพธ์จากบล็อกเป็นเท็จ ในขณะที่เมสเซจ select: ผลลัพธ์จากบล็อกเป็นจริง

- เมสเซจ collect:

ตัวอย่างเช่น #(1 13 7 10) collect:[:ili*i]

การทำงานของเมสเซจ collect: มีผลทำให้บล็อกของโคดทำงานสำหรับแต่ละสมาชิกของรีซีพเวอร์ และให้ผลลัพธ์เป็นกลุ่มของผลลัพธ์ทั้งหมดที่ส่งกลับโดยบล็อก

- **การส่งค่ากลับของฟังก์ชัน** ในสมอลทอคใช้เครื่องหมาย ^(caret) นำหน้านิพจน์ซึ่งเป็นผลลัพธ์ของเมธธอด เมธธอดจะหยุดการทำงานและส่งค่าผลลัพธ์ เช่น

```
^ answer
```

เปรียบเทียบกับภาษาปาสคาลผลลัพธ์ของฟังก์ชันจะเขียนเป็น

```
functionName:=answer;
return
```

2.2.4 ตัวอย่างโปรแกรม

```
| s c f k |
f:=Array new:26.
s:=Prompter prompt:'Enter line' default:''.
1 to: 26 do:[:i|f at:i put:0].
1 to: s size do:[:i| c:=(s at:i) asLowerCase.
                    c isLetter
                        ifTrue:[k:=c asciiValue - $a asciiValue + 1.
                                f at:k put:(f at:k) + 1]].
^f
```

2.3 การใช้สมอลทอคไฟว์ (Smalltalk/v)

สมอลทอคไฟว์คือระบบแบบเมนู ดังนั้นคุณสามารถใช้คำสั่งเพื่อเลือกสิ่งต้องการได้จากเมนูโดยการใช้เมาส์ (mouse) หรือกดคีย์บอร์ด

1) การเรียกเมนู(pop up menu)

โดยการคลิกปุ่มขวาของเมาส์ หรือกดคีย์บอร์ดปุ่ม 'Del'

2) การเลือกสิ่งที่อยู่ในเมนู

เลื่อนเคอร์เซอร์ (cursor) ไปที่สิ่งที่ต้องการเลือกแล้วกดปุ่มซ้ายของเมาส์หรือกดคีย์บอร์ดปุ่ม '+' บนคีย์บอร์ดด้านตัวเลข (numeric keypad)

2.3.1. เมนู

สมอลทอคไฟว์ประกอบด้วยเมนูหลายชนิด คือ

1) เมนูระบบ (System Menu)

เป็นเมนูที่เราต้องใช้บ่อยๆ การเรียกเมนูระบบทำได้โดยการเลื่อนเคอร์เซอร์ออกนอกหน้าต่าง (window) ไปที่พื้นจอภาพ (background) แล้วกดปุ่มขวาของเมาส์ หรือกดคีย์บอร์ดปุ่ม 'Del' หน้าต่างของเมนูระบบจะเปิดขึ้นใหม่และกระทำฟังก์ชันระดับระบบเช่น ออกจากระบบ วาดจอภาพใหม่ เป็นต้น

2) วินโดว์เมนู (Window Menu)

การเรียกวินโดว์เมนูทำได้โดยการเลื่อนเคอร์เซอร์ไปที่ป้ายของวินโดว์หรือขอบของวินโดว์ แล้วกดปุ่มขวาของเมาส์ หรือกดคีย์บอร์ดปุ่ม 'Del' ออกจากวินโดว์เมนูโดยการเลื่อนเคอร์เซอร์ออกจากวินโดว์เมนู วินโดว์เมนูของเมนูที่แตกต่างกันก็จะมีรายละเอียดที่แตกต่างกันด้วย

3) เพนเมนู(Pane menu)

การเรียกเพนเมนูทำได้โดยการเลื่อนเคอร์เซอร์ไปที่จุดใดจุดหนึ่งของวินโดว์เพน แล้วกดปุ่มขวาของเมาส์หรือกดคีย์บอร์ดปุ่ม 'Del' ซึ่งจะมีรายละเอียดแตกต่างกันสำหรับเมนูแต่ละชนิด

เราสามารถเปิดวินโดว์ได้พร้อมๆกันหลายวินโดว์ เมื่อต้องการใช้วินโดว์ใดก็เลื่อนเคอร์เซอร์เข้าไปในขอบเขตของวินโดว์นั้น แล้วกดปุ่มซ้ายของเมาส์หรือกดคีย์บอร์ดปุ่ม '+' บนคีย์บอร์ดด้านตัวเลข ป้ายของวินโดว์จะเปลี่ยนเป็นสีเข้มขึ้นเพื่อแสดงว่าเป็นแอคทีพวินโดว์ (active window)

ถ้าต้องการเพิ่มข้อความบนเพนของวินโดว์ให้เลื่อนเคอร์เซอร์ไปที่ตำแหน่งที่ต้องการแล้วกดปุ่มซ้ายของเมาส์หรือกดคีย์บอร์ดปุ่ม '+' บนคีย์บอร์ดด้านตัวเลขจะปรากฏเครื่องหมายที่ตำแหน่งนั้นเรียกว่าจุดเพิ่ม (insert point) เราสามารถพิมพ์สิ่งที่ต้องการเพิ่มเข้าไปได้

ถ้าต้องการลบข้อความบางอย่างออกจะใช้วิธีคล้ายๆการเพิ่มคือเลื่อนเคอร์เซอร์ไปที่หลังข้อความนั้นแล้วกดปุ่มซ้ายของเมาส์หรือกดปุ่มคีย์ '+' บนคีย์บอร์ดด้านตัวเลข แล้วใช้คีย์ลบถอยหลัง (backspace) ลบข้อความที่ต้องการ

ถ้าต้องการเลือกข้อความใดข้อความหนึ่งให้เลื่อนเคอร์เซอร์ไปที่ตัวเริ่มต้นของข้อความกดปุ่มซ้ายของเมาส์ค้างไว้ และลากเมาส์ไปที่ท้ายสุดของข้อความแล้วจึงปล่อยปุ่มซ้ายของเมาส์ จะปรากฏเป็นแถบสว่างบนข้อความที่เลือก แต่ถ้าใช้คีย์บอร์ดให้กดปุ่ม '+' บนคีย์บอร์ดด้านตัวเลขที่จุดเริ่มต้นของข้อความ และกดปุ่ม '-' บนคีย์บอร์ดด้านตัวเลขที่ท้ายสุดของข้อความ

2.3.2. ออบเจคและเมสเซจ

หลักการพื้นฐานของภาษาสมอลทอคคือออบเจคและเมสเซจ ออบเจคคือบล็อกพื้นฐานของสมอลทอค ตัวอย่างเช่น

1234 คือเลขจำนวนเต็ม

$A คือตัวอักษรเอ

#(1 2 3) คืออาร์เรย์ของเลขจำนวนเต็ม ซึ่งมีอยู่ด้วยกัน 3 ออบเจค

#('array' 'of' 5 'string' 'and' '2') คืออาร์เรย์ของออบเจคที่ต่างชนิดและต่างขนาด

#(1 ('two' 'three') 4) คืออาร์เรย์ซึ่งมี 3 ออบเจค และออบเจคที่ 2 คืออาร์เรย์ ซึ่งแน่นอนที่สุดว่าออบเจคไม่สามารถทำอะไรได้ด้วยตัวเอง ในสมอลทอคเราส่งเมสเซจไปยังออบเจคเพื่อทำให้เกิดสิ่งหนึ่งขึ้นมา เมสเซจนั้นจะทำนองเดียวกับการเรียกฟังก์ชันในภาษาอื่น ตัวอย่างเช่น 20 factorial คือการส่งเมสเซจ factorial ไปยังออบเจค 20

เมสเซจประกอบด้วย

1) รีซีพเวอร์ออบเจค (Receiver Object)

2) เมสเซจซีเลคเตอร์ (Message Selector)

3) อาร์กิวเมนต์ ซึ่งอาจจะมีหรือไม่มีก็ได้

ตัวอย่างเช่น 'Now is the time' size ข้อความเป็นรีซีพเวอร์ออบเจค และ size คือเมสเซจซีเลคเตอร์ซึ่งไม่มีอาร์กิวเมนต์ เมสเซจที่ไม่มีอาร์กิวเมนต์เรียกว่ายูนารี่เมสเซจ (Unary Message) เมสเซจที่มีหนึ่งอาร์กิวเมนต์ หรือมากกว่าหนึ่งอาร์กิวเมนต์เรียกว่าคีย์เวิร์ดเมสเซจ (Keyword Message)

การคำนวณในภาษาสมอลทอคมองดูคล้ายกับภาษาอื่นๆ โดยทั่วไป แต่ความจริงคือเมสเซจตัวอย่างเช่น 3 + 4 ตัวเลข 3 คือรีซีพเวอร์ออบเจค และ + คือเมสเซจซีเลคเตอร์ ซึ่งมีเลข 4 เป็นอาร์กิวเมนต์ โดยที่เมสเซจคล้ายฟังก์ชันดังนั้นจึงส่งค่ากลับเป็นออบเจค เช่น 'Hello'size + 4 เมสเซจ size ส่งค่ากลับเป็นออบเจคตัวเลขและเป็นรีซีพเวอร์ออบเจคของเมสเซจ + ซึ่งมี 4 เป็นอาร์กิวเมนต์ การแทนเมสเซจมักจะเป็นชุดเพื่อสั่งให้ทำงานใดๆ เช่น

```
Turtle blank.
Turtle home.
Turtle go:100.
Turtle turn:120.
```

ในกรณีที่เมสเซจส่งถึงออบเจคเดียวกันเราสามารถเขียนแบบย่อได้เรียกว่า เคสเคดเมสเซจ (cascaded message) ซึ่งแต่ละเมสเซจจะเขียนคั่นด้วยเครื่องหมาย ; ตัวอย่างเช่น

```
Turtle blank;
    home;
    go:100;
    turn:120;
    go:100;
    turn:120;
    go:100;
    turn:120.
```

ออบเจคมีสเตทซึ่งออบเจคสามารถจำสิ่งๆได้ เมสเซจเปลี่ยนสเตทของออบเจค จากตัวอย่างของออบเจค Turtle จำตำแหน่งของมันเอง หัวเรื่อง และสีได้ เมสเซจ black และ white เปลี่ยนสีของออบเจค Turtle เมสเซจ go: และ home เปลี่ยนตำแหน่งของมัน และเมสเซจ turn: เปลี่ยนหัวเรื่อง

2.3.3. คลาสและเมธธอด

แนวความคิดของคลาสและเมธธอดทำให้สมอลทอคไม่เหมือนโปรแกรมภาษาอื่น

1) คลาส

การแก้ไขปัญหาโดยสมอลทอคนั้นรวมการจัดประเภทของออบเจคซึ่งขึ้นกับความคล้ายกันและความแตกต่างกัน คลาสกำหนดพฤติกรรมของออบเจคแบบเดียวกัน โดยการกำหนดตัวแปรและเมธธอดภายในออบเจค ซึ่งเมธธอดสามารถตอบกับเมสเซจที่ส่งมายังออบเจคได้

ทุกๆ ออบเจคคืออินสแตนซ์ (instance) หรือสมาชิกของคลาส ตัวอย่าง เช่น #(1 2 3) และ #(sam joe) คืออินสแตสซ์ของคลาสอาร์เรย์ แต่ 'north' และ 'south' คืออินสแตสซ์ของคลาสสตริงค์ ทุกออบเจครู้ว่าตนเป็นสมาชิกคลาสใด ตัวแปรในออบเจคเรียกว่าตัวแปรอินสแตนซ์

2) เมธธอด

เมธธอดคือ สมอลทอคโคด อะกอริทึมที่อธิบายพฤติกรรมของออบเจค และการกระทำของออบเจค เมธธอดคล้ายกับฟังก์ชันในภาษาอื่น ๆ เมื่อเมสเซจส่งถึงออบเจคเมธธอดจะถูกกระทำ และส่งผลลัพธ์กลับเป็นออบเจค

บทที่ 3
ระบบงานบริหารพัสดุสายซ่างอากาศ

3.1 ระบบงานบริหารพัสดุสายซ่างอากาศ

ระบบงานบริหารพัสดุสายซ่างอากาศเป็นระบบงานที่ใช้ควบคุมการเบิกจ่ายพัสดุซึ่งเกี่ยวกับการบินทั้งหมด [7] ประกอบด้วยขั้นตอนการทำงานดังแสดงในดาต้าโฟลไดอะแกรมภาพที่ 24

3.2 โพรเซส 1.0 การรับเอกสาร

ฝ่ายรับเอกสารตรวจสอบความถูกต้องของเอกสารตามโพรเซส 1.1 ถ้าเอกสารไม่ถูกต้องจะส่งคืนเอกสารตามโพรเซส1.4 ส่งคืนเอกสารกลับคืนไปยังหน่วยที่เป็นเจ้าของเอกสาร ถ้าเอกสารถูกต้องใส่เลขที่รับเอกสารตามโพรเซส1.2 แยกประเภทของเอกสารตามโพรเซส1.3 ถ้าเป็นเอกสารเบิกจะส่งต่อไปยังโพรเซส7.0 ถ้าเป็นเอกสารส่งคืนจะส่งต่อไปยังโพรเซส2.0 ถ้าเป็นเอกสารรับพัสดุจัดหาจะส่งต่อไปยังโพรเซส9.0 หรือถ้าเป็นเอกสารรับพัสดุจากการซ่อมจะส่งต่อไปยังโพรเซส5.0 ดังแสดงในภาพที่ 25

3.3 โพรเซส 2.0 การรับพัสดุส่งคืน

หน่วยผู้ใช้ส่งคืนพัสดุที่เกินความจำเป็นของหน่วยให้กับคลังใหญ่ หรือในกรณีที่คลังใหญ่เรียกพัสดุคืนเพื่อจ่ายให้กับหน่วยผู้ใช้ที่มีความจำเป็นเร่งด่วนกว่า เช่น เครื่องบินขึ้นบินไม่ได้เป็นต้น เจ้าหน้าที่พัสดุจะตรวจสอบสภาพของพัสดุที่ส่งคืนว่าเป็นพัสดุดีหรือพัสดุชำรุดตามโพรเซส2.1 ถ้าเป็นพัสดุดีเจ้าหน้าที่เพิ่มยอดของพัสดุดีตามโพรเซส2.2 และส่งต่อไปยังโพรเซส 3.0 เพื่อดำเนินการทำหักล้างค้างจ่ายพัสดุต่อไป ถ้าเป็นพัสดุชำรุดเพิ่มยอดจำนวนพัสดุชำรุดตามโพรเซส2.3 และส่งพัสดุชำรุดเข้าไปในโพรเซส 4.0 เพื่อดำเนินการส่งซ่อมพัสดุต่อไปดังแสดงในภาพที่ 26

3.4 โพรเซส 3.0 การหักล้างค้างจ่ายพัสดุ

รายการรับพัสดุดีจากแหล่งการจัดหา หรือจากหน่วยซ่อมพัสดุจะถูกตรวจสอบว่ามีรายการค้างจ่ายพัสดุหมายเลขนี้ในบัญชีค้างจ่ายหรือไม่ตามในโพรเซส 3.1 ถ้าไม่มีส่งพัสดุไปเก็บในคลังพัสดุตามโพรเซส 3.2 ถ้ามีรายการค้างจ่ายพัสดุเลือกรายการค้างจ่ายพัสดุ โดยจะพิจารณา

ว่าหน่วยใดเบิกก่อนหรือหน่วยใดมีความจำเป็นเร่งด่วนตามในโพรเซส 3.3 เมื่อเลือกรายการค้างจ่ายได้แล้วจะลบรายการค้างจ่ายนั้นออกจากบัญชีค้างจ่ายตามโพรเซส 3.4 ลดยอดพัสดุดีในบัญชีคุมพัสดุตามโพรเซส 3.5 ทำการจ่ายพัสดุตามโพรเซส 3.6 ถ้ามีพัสดุเหลือจะส่งพัสดุส่วนที่เหลือนั้นเข้าไปเก็บในคลังพัสดุตามโพรเซส 3.7 ดังแสดงในภาพที่ 27

3.5 โพรเซส 4.0 การส่งซ่อมพัสดุ

รายการพัสดุชำรุดจากโพรเซส 2.0 ทำรายการรอรับงานซ่อมตามโพรเซส 4.1 ลดจำนวนพัสดุชำรุดและเพิ่มจำนวนพัสดุรอรับงานซ่อมตามโพรเซส 4.2 และส่งพัสดุส่งไปซ่อมตามประเภทของพัสดุ เช่น ซ่อมในประเทศหรือส่งซ่อมต่างประเทศดังแสดงในรูปที่ 28

3.6 โพรเซส 5.0 การรับพัสดุจากซ่อม

รายการรับพัสดุจากการซ่อมซึ่งได้มาจากโพรเซส 1.0 ลบรายการพัสดุรอรับงานซ่อมออกจากบัญชีรอรับงานซ่อมตามโพรเซส 5.1 ตรวจสภาพของพัสดุที่รับตามโพรเซส 5.2 ถ้าเป็นพัสดุดีเพิ่มจำนวนพัสดุดีในบัญชีคุมพัสดุตามโพรเซส 5.3 แล้วส่งต่อไปยังโพรเซส 3.0 ถ้าเป็นพัสดุชำรุดจะเพิ่มจำนวนพัสดุชำรุดในบัญชีคุมตามโพรเซส 6.0 ดังแสดงในภาพที่ 29

3.7 โพรเซส 6.0 การจำหน่ายพัสดุ

รายการพัสดุชำรุดจากโพรเซส 5.0 ลดจำนวนพัสดุชำรุดในบัญชีคุมพัสดุ ตามโพรเซส 6.1 เพิ่มจำนวนพัสดุจำหน่ายในบัญชีคุมพัสดุตามโพรเซส 6.2 และส่งไปเก็บในคลังจำหน่ายพัสดุเพื่อรอการจำหน่ายต่อไป ดังแสดงในภาพที่ 30

3.8 โพรเซส 7.0 การจ่ายพัสดุ

เอกสารเบิกพัสดุจากโพรเซส 1.0 ตรวจสอบพัสดุเพื่อหาประเภทของพัสดุว่าเป็นพัสดุประเภทพัสดุซ่อมหมุนเวียนหรือพัสดุสิ้นเปลืองตามโพรเซส 7.1 ถ้าเป็นพัสดุซ่อมหมุนเวียนจะตรวจสอบว่ามีพัสดุชำรุดมาแลกเปลี่ยนจึงจะขอเบิกพัสดุใหม่ได้ตามโพรเซส 7.7 ถ้าไม่มีพัสดุส่งคืนจะส่งเอกสารเบิกกลับคืนหน่วยผู้ใช้ตามโพรเซส7.8 แต่ถ้ามีพัสดุส่งคืนจะตรวจสอบยอดพัสดุว่ามีจำนวนเพียงพอที่จะจ่ายหรือไม่ตามโพรเซส 7.2 ถ้ามีจำนวนพัสดุเพียงพอที่จะจ่ายจะลดจำนวนพัสดุในบัญชีคุมตามโพรเซส 7.3 สั่งจ่ายพัสดุตามโพรเซส 7.4 ถ้ามีพัสดุไม่พอจ่ายจะตรวจสอบว่ามีพัสดุที่สามารถใช้แทนกันได้หรือไม่ เพื่อนำมาพิจารณาจ่ายแทนพัสดุที่เบิกตามโพร

เซส 7.5 แต่ถ้าไม่สามารถที่จะจ่ายพัสดุได้จะทำรายการพัสดุค้างจ่ายไว้ตามโพรเซส 7.6 ส่งรายการค้างจ่ายเข้าไปยังโพรเซส8.0 ต่อไป ดังแสดงในภาพที่ 32

3.9 โพรเซส 8.0 การจัดหาพัสดุ

ฝ่ายจัดหาคำนวณหาจำนวนพัสดุที่ต้องจัดหาเพิ่มเติม เนื่องจากพัสดุมีจำนวนมากกว่าแสนรายการ แต่งบประมาณมีจำกัด ดังนั้นในทางปฏิบัติจึงใช้วิธีเลือกรายการพัสดุที่มีจำนวนพัสดุต่ำกว่าจุดเบิก(reorder point) มาคำนวณหาจำนวนพัสดุที่องจัดหาเพิ่มเติม การคำนวณมีดังนี้

จำนวนพัสดุที่ใช้ต่อวัน = จำนวณพัสดุที่ใช้ไปทั้งหมด/จำนวณวัน

[DDR=(consumption1+consumption2+consumption3) / (365x3)

โดย DDR : Daily Demand Rate ,consumption: ความสิ้นเปลืองพัสดุ]

จำนวนพัสดุคงคลัง = จำนวณพัสดุที่ใช้ต่อวัน x 365 [stock-level = DDR x 365]

จำนวนพัสดุที่จุดเบิก = จำนวณพัสดุที่ใช้ต่อวัน x 270 [reorder-point = DDR x 270]

จำนวณพัสดุที่จุดวิกฤติ = จำนวณพัสดุที่ใช้ต่อวัน x 90 [safety-level = DDR x 90]

requirment = (stock-level + dueout) - (onhand + duein + workorder)

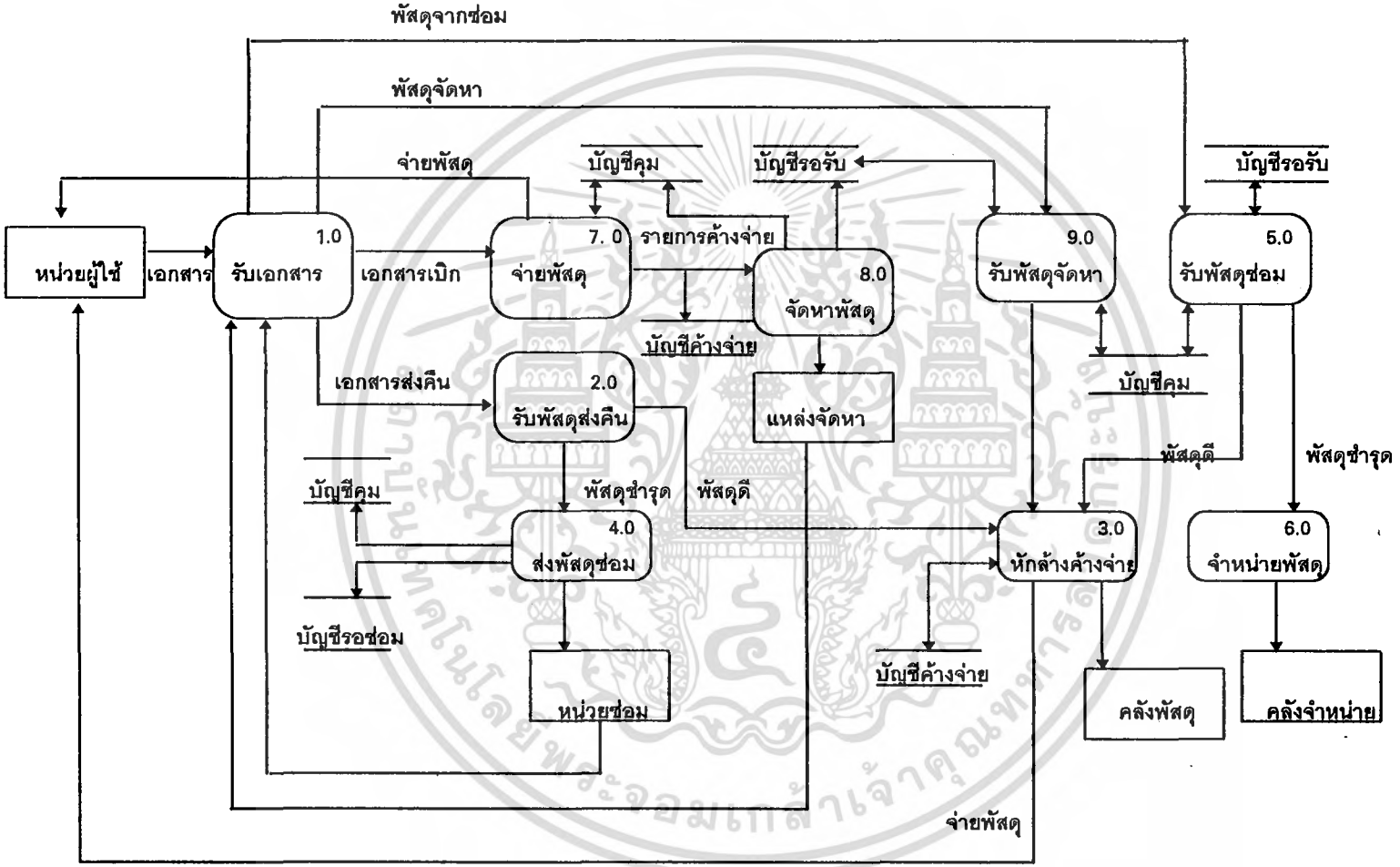
โดย จำนวนพัสดุที่ต้องมี : จำนวนพัสดุคงคลังรวมกับจำนวนพัสดุที่ค้างจ่าย

จำนวนพัสดุที่มีอยู่: จำนวนพัสดุดีรวมกับจำนวนพัสดุที่รอรับจากการจัดหา และจากการซ่อม

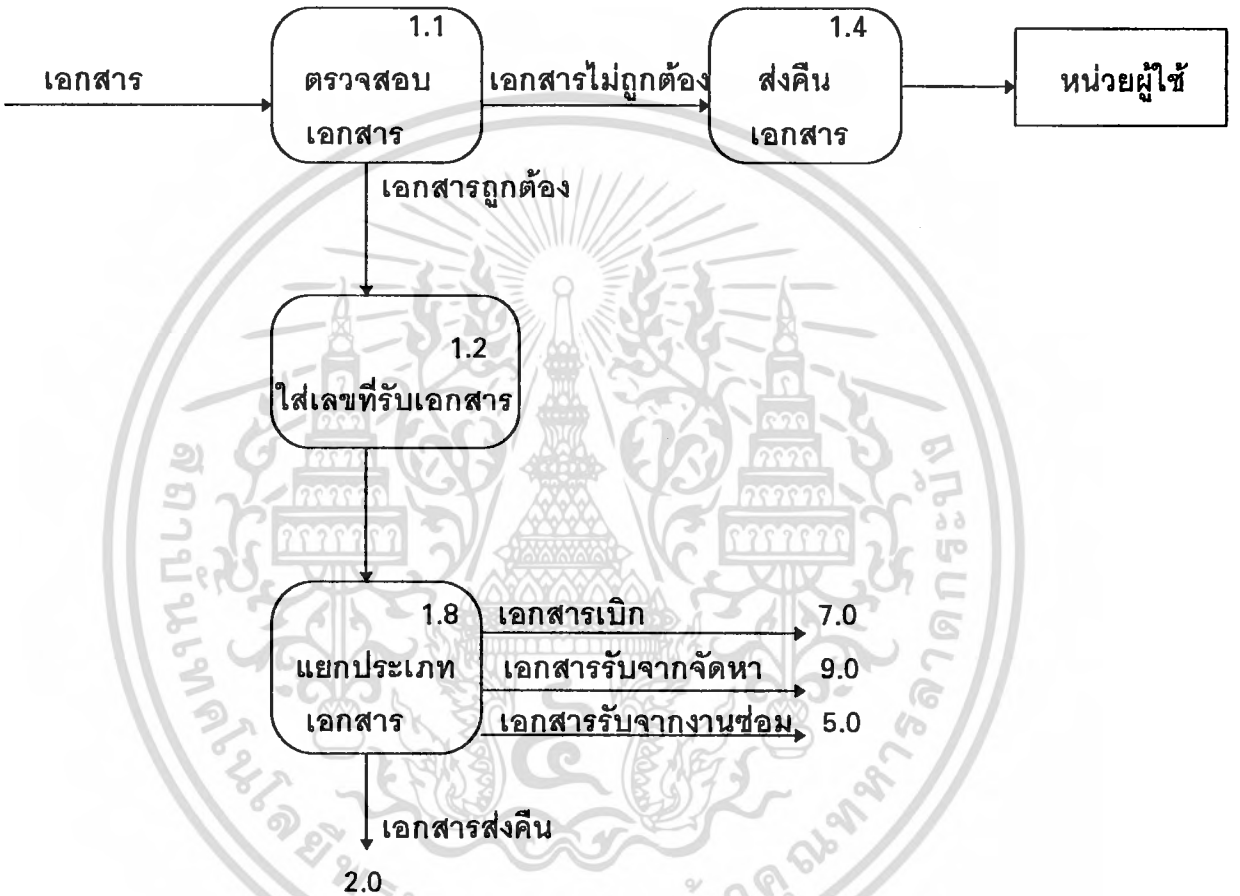
การคำนวณกำหนดไว้ในโพรเซส 8.1 ทำรายการรอรับการจัดหาในบัญชีรอรับการจัดหาตามโพรเซส 8.2 ดำเนินการจัดหาตามโพรเซส 8.3 ส่งเอกสารการจัดหาไปที่แหล่งจัดหา ดังแสดงในภาพที่ 31

3.10 โพรเซส 9.0 การรับพัสดุจากการจัดหา

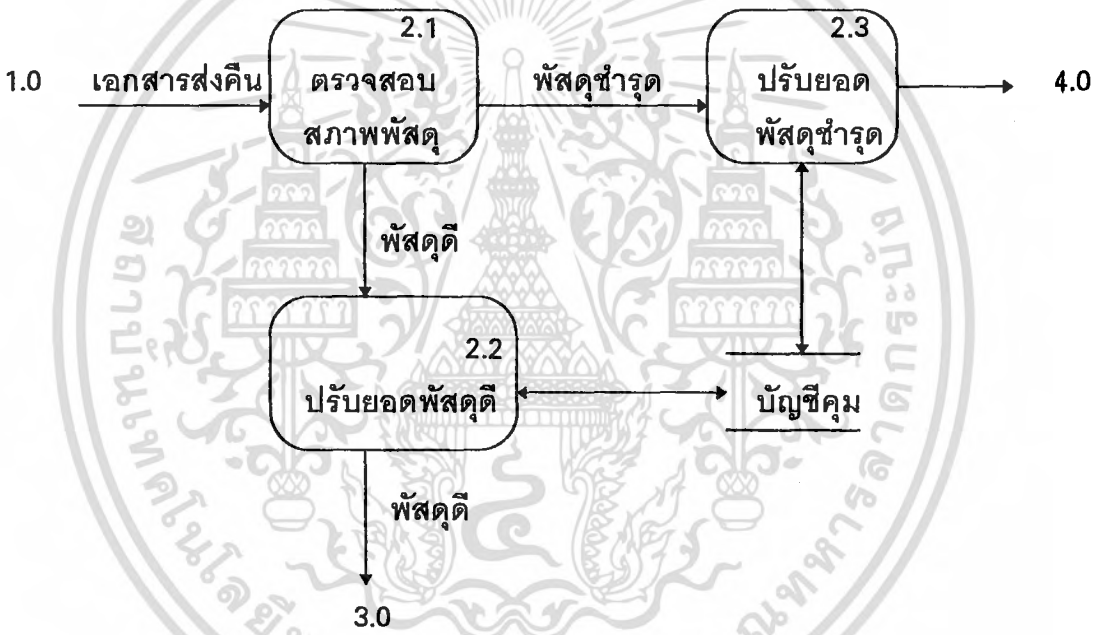
รายการรับพัสดุจากการจัดหาจากโพรเซส 1.0 ลบรายการรอรับพัสดุจากการจัดหาในบัญชีรอรับพัสดุจากการจัดหาตามโพรเซส 9.1 เพิ่มจำนวนพัสดุในบัญชีคุมพัสดุตามโพรเซส 9.2 และส่งรายการพัสดุดีเข้าไปยังโพรเซส 3.0 ดังแสดงในภาพที่ 33



ภาพที่ 24 ดาต้าโฟลไดอะแกรมของระบบงาน ระดับ 0

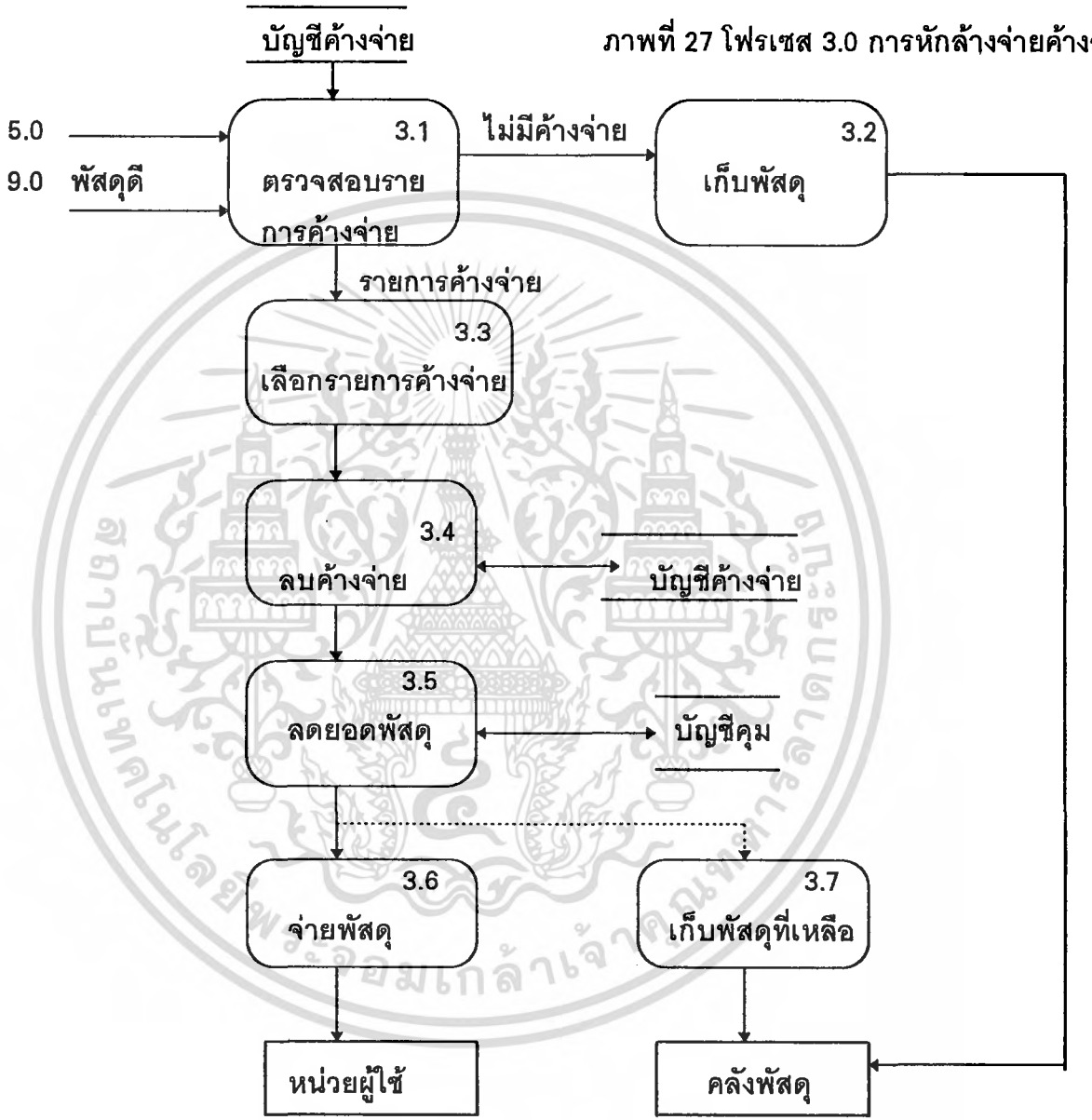


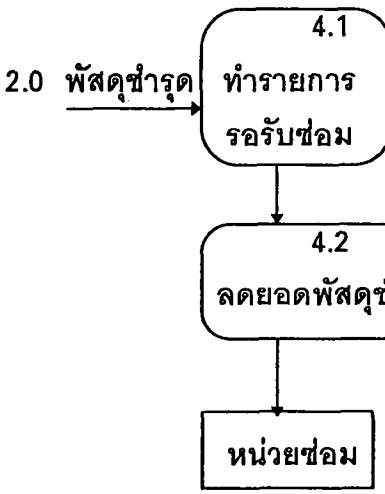
ภาพที่ 25 โฟรเซส 1.0 การรับเอกสาร ระดับ 1



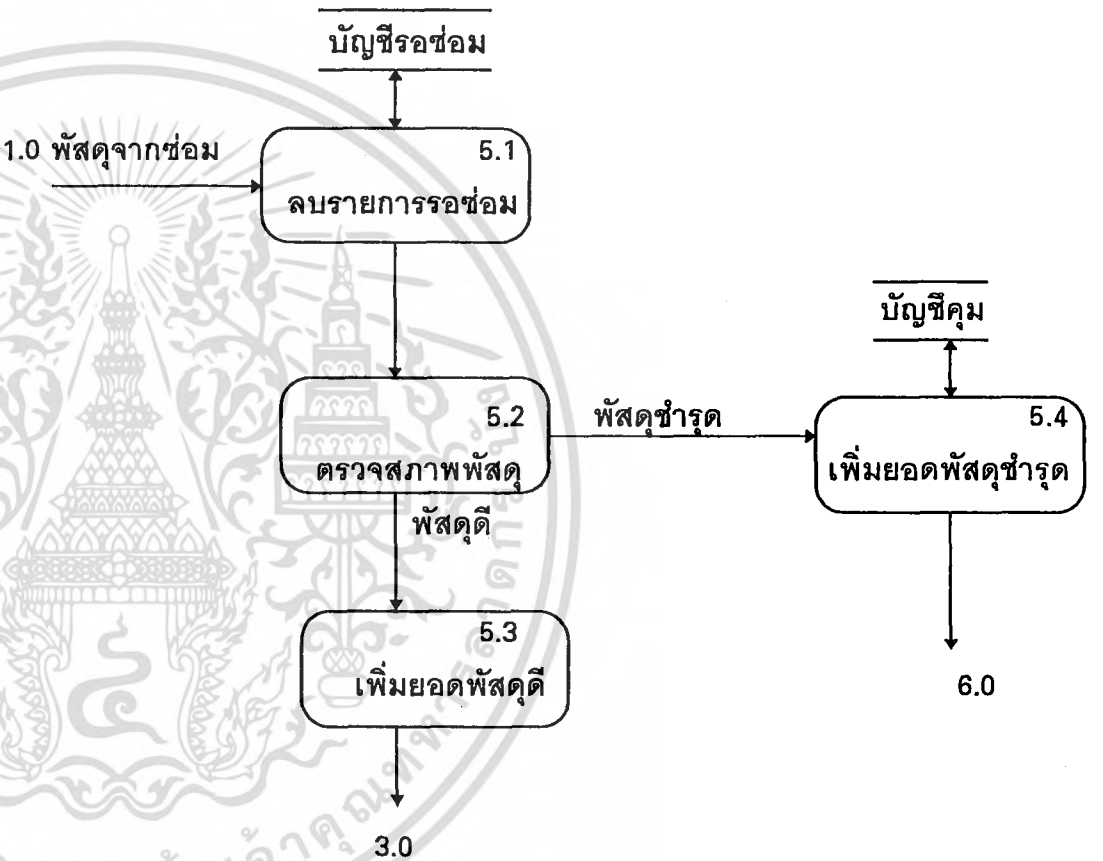
ภาพที่ 26 โฟรเซส 2.0 การรับพัสดุส่งคืน ระดับ 1

ภาพที่ 27 โฟรเซส 3.0 การหักล้างจ่ายค้างจ่าย ระดับ 1

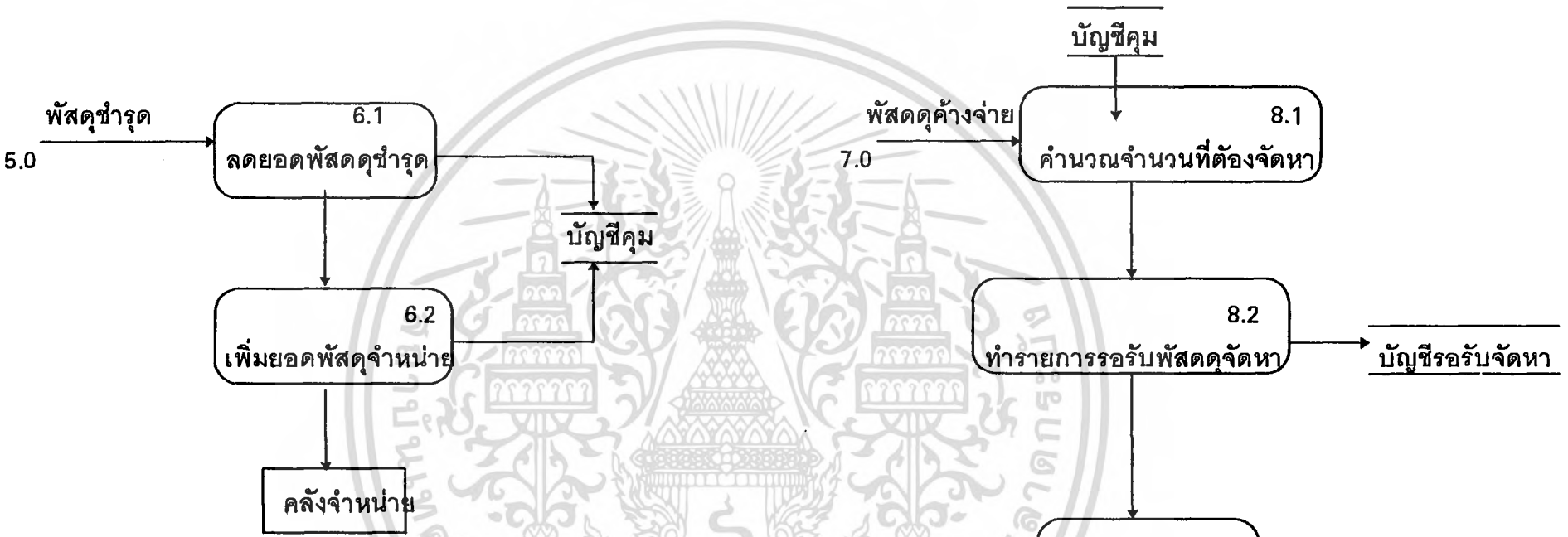




ภาพที่ 28 โพรเซส 4.0 ส่งซ่อมพัสดุ ระดับ 1



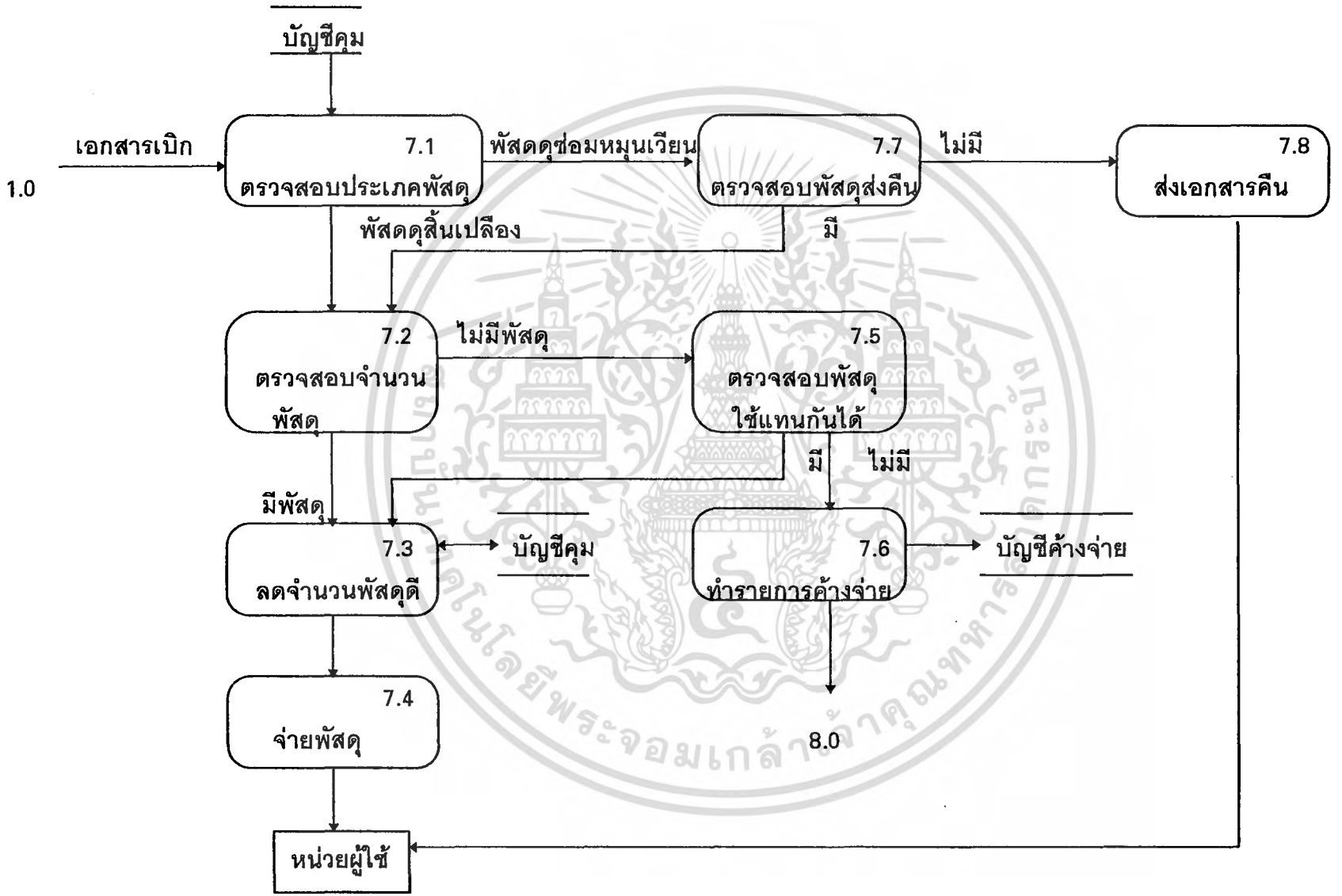
ภาพที่ 29 โพรเซส 5.0 การรับพัสดุจากซ่อม ระดับ 1

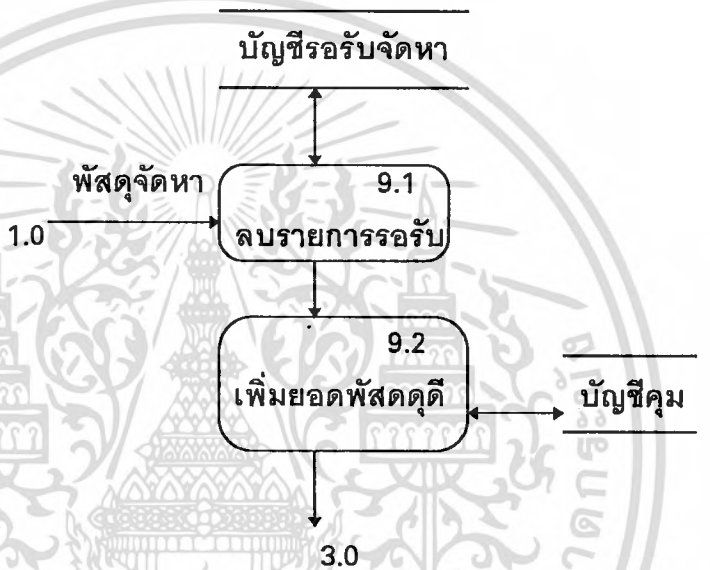


ภาพที่ 30 โพรเซส 6.0 การจำหน่ายพัสดุ ระดับ 1

ภาพที่ 31 โพรเซส 8.0 การจัดหาพัสดุ ระดับ 1

ภาพที่ 32 โพรเซส 7.0 การจ่ายพัสดุ





ภาพที่ 33 โฟรเซส 9.0 การรับพัสดุจัดหา ระดับ 1

บทที่ 4
การพัฒนาระบบงาน

4.1 การวิเคราะห์เชิงวัตถุ

ขั้นตอนในการวิเคราะห์เชิงวัตถุมีดังต่อไปนี้คือ

1) การกำหนดประโยคปัญหา (Problem Statement) ซึ่งประโยคปัญหานี้อาจจะยังไม่สมบูรณ์หรือไม่อยู่ในรูปแบบฟอร์มที่ถูกต้อง ดังนั้นการวิเคราะห์จะต้องทำให้ประโยคปัญหานี้สมบูรณ์ ถูกต้องครบถ้วน ไม่คลุมเครือ และมีความคงเส้นคงวา (consistent) ประโยคปัญหาไม่ได้เป็นสิ่งที่ไม่มีวันเปลี่ยนแปลงแต่เป็นพื้นฐานที่สามารถแต่งเติมความต้องการที่แท้จริงได้ จนในที่สุดประโยคปัญหานี้สามารถใช้อธิบายระบบได้

2) การสร้างแบบจำลองของการวิเคราะห์ (Analysis Model) แบบจำลองนี้ต้องสามารถใช้แทนประโยคปัญหาได้อย่างครบถ้วนแม่นยำ สั้นกระทัดรัด แต่จุใจความ ซึ่งทำให้หาคำตอบและวิธีการแก้ปัญหาจากแบบจำลองปัญหานั้นได้ การสร้างแบบจำลองภายในขอบเขตของปัญหา ทำให้ผู้พัฒนาระบบสามารถแก้ไขความเข้าใจที่อาจจะผิดพลาดได้ตั้งแต่ในระยะแรกๆ ของการสร้างระบบงาน ซึ่งยังสามารถที่จะแก้ไขได้ง่ายอยู่

4.2 ประโยคปัญหา (Problem Statement)

ขั้นตอนแรกของการสร้างระบบงานคือการกำหนดความต้องการ ประโยคของความต้องการจะประกอบด้วย ขอบเขตของปัญหา อะไรคือสิ่งที่ต้องการ รายละเอียดของระบบงาน สมมุติฐาน และความต้องการเกี่ยวกับการทำงาน ประโยคปัญหาต้องกำหนดว่าจะทำอะไรไม่ใช่จะทำอย่างไร นั่นคือเป็นประโยคของความต้องการไม่ใช่เป็นสมมุติฐานของการแก้ปัญหา คู่มือการใช้ระบบงานที่ต้องการสร้างคือประโยคปัญหาที่ดี เจ้าของระบบงานควรระบุว่าอะไรที่จะต้องมีในระบบงานและอะไรคือสิ่งที่ให้เลือกได้ เพื่อหลีกเลี่ยงการออกแบบระบบที่มีข้อบังคับมากเกินไป เจ้าของระบบงานไม่ควรอธิบายรายละเอียดภายในระบบงานมากจนเกินไป เพราะว่าจะทำให้การสร้างระบบขึ้นมาใช้งานจริงไม่มีความยืดหยุ่นเท่าที่ควร ข้อกำหนดการทำงานและรูปแบบของการติดต่อเพื่อโต้ตอบกับภายนอกระบบคือความต้องการที่เป็นไปตามทฤษฎี ประโยคปัญหาหลายประโยคจากเอกชน บริษัท และหน่วยงานของรัฐ รวมเข้าด้วยกันเป็นความต้องการที่แท้

จริงสำหรับการออกแบบระบบ ในบางครั้งอาจมีเหตุผลที่บังคับว่าต้องการเครื่องคอมพิวเตอร์ชนิดนี้ หรือต้องใช้ภาษาคอมพิวเตอร์ภาษานี้หรือมีความจำเป็นที่จะต้องใช้อะกอริทึม (algorithm) แบบนี้ นักวิเคราะห์ระบบจะต้องแยกความต้องการที่แท้จริงและความต้องการที่ใช้สร้างระบบ นักวิเคราะห์ระบบจะต้องมองความต้องการแฝงเหล่านั้นเป็นข้อจำกัดของความยืดหยุ่น อาจจะมีเหตุผลที่เป็นนโยบายหรือเป็นระเบียบการจัดองค์การซึ่งเป็นความต้องการแฝงแต่ในที่สุดนักวิเคราะห์ระบบต้องรู้ว่าสิ่งเหล่านี้ไม่ใช่ลักษณะที่สำคัญของปัญหา ประโยคปัญหาอาจจะมีรายละเอียดมากหรือมีรายละเอียดน้อย ตัวอย่างเช่นโปรแกรมบัญชีเงินเดือน หรือระบบการเงินต้องการรายละเอียดมาก ส่วนงานวิจัยสิ่งใหม่ๆ ไม่ต้องการรายละเอียดมาก แต่ต้องกำหนดจุดมุ่งหมายที่ชัดเจน ประโยคปัญหาส่วนมากมักจะคลุมเครือ ไม่สมบูรณ์ หรือไม่คงเส้นคงวา ความต้องการบางอย่างผิดอย่างชัดเจน ความต้องการบางอย่างกำหนดถูกต้องแต่ผลลัพธ์ที่ได้ไม่ใช่พฤติกรรมของระบบ หรือค่าใช้จ่ายในการสร้างระบบไม่สมเหตุสมผล ความต้องการบางอย่างดูเหมือนว่ามีเหตุผลในครั้งแรกแต่กลับทำงานออกมาแล้วไม่ตรงกับที่ต้องการหรือที่คิดไว้ ประโยคปัญหาคือจุดเริ่มต้นของการเข้าใจปัญหาไม่ใช่เป็นเอกสารที่ไม่มีวันเปลี่ยนแปลง จุดมุ่งหมายของการวิเคราะห์คือการเข้าใจปัญหาและการสร้างระบบขึ้นมาอย่างครบถ้วน ไม่มีเหตุผลที่จะคิดว่าการจัดเตรียมประโยคปัญหาจะถูกต้องโดยไม่ต้องมีการวิเคราะห์ นักวิเคราะห์ระบบต้องทำงานร่วมกับเจ้าของระบบงานเพื่อขัดเกลาความต้องการจนกระทั่งสามารถใช้แทนความต้องการที่แท้จริงของเจ้าของระบบงานได้

ประโยคปัญหาของงานวิจัยในที่นี้ก็คือระบบงานบริหารพัสดุสายช่างอากาศที่ได้กล่าวถึงรายละเอียดของระบบงานที่ต้องการไว้ในบทที่ 3

4.3 ออบเจคโมเดลลิ่ง (Object Modeling)

ขั้นตอนแรกในการวิเคราะห์ความต้องการคือการสร้างออบเจคโมเดล ซึ่งแสดงโครงสร้างของข้อมูลแบบคงที่ของระบบ และแบ่งออกเป็นการทำงานส่วนย่อยๆ ออบเจคโมเดลใช้อธิบายคลาส และความสัมพันธ์ระหว่างคลาส สิ่งที่สำคัญที่สุดคือการจัดระเบียบในระดับบนสุดของระบบ เป็นคลาส ซึ่งเชื่อมโยงกันด้วยแอสโซซิเอชัน (association) การแบ่งในระดับต่ำกว่าภายในคลาสมีความสำคัญน้อยกว่า ออบเจคโมเดลจะถูกสร้างขึ้นก่อนไดนามิกโมเดล (dynamic model) และฟังก์ชันนัลโมเดล (functional model) เพราะว่าออบเจคโมเดลสามารถกำหนดได้ดีกว่าขึ้นกับรายละเอียดในระบบงานน้อยกว่า มีความคงที่ขณะแก้ปัญหามากกว่า และสามารถทำความเข้าใจได้ง่ายกว่าข่าวสารสำหรับออบเจคโมเดลได้มาจากประโยคปัญหา ผู้เชี่ยวชาญระบบงาน ความ

รู้ทั่วๆไป ถ้าผู้ออกแบบระบบไม่ใช่ผู้เชี่ยวชาญในเรื่องนั้นข่าวสารต้องได้รับมาจากผู้เชี่ยวชาญระบบงาน และต้องตรวจสอบแบบจำลองซ้ำอีก ออบเจคโมเดลช่วยให้นักคอมพิวเตอร์และผู้เชี่ยวชาญระบบงานได้มีการติดต่อสื่อสารกัน อันดับแรกกำหนดคลาสและแอสโซซิเอชันตามที่มีอิทธิพลเหนือโครงสร้างทั้งหมดและเข้าใกล้กับปัญหา ต่อไปเพิ่มแอทริบิวท์ (attribute) เพื่อใช้อธิบายเนตเวิร์คขั้นพื้นฐานของคลาสและแอสโซซิเอชัน ต่อจากนั้นรวบรวมและจัดระเบียบของคลาสโดยใช้อินเฮริทันซ์ (inheritance) การพยายามจะกำหนดอินเฮริทันซ์โดยตรงโดยไม่มีการกำหนดคลาสและแอทริบิวท์ของคลาสในระดับต่ำมาก่อนมักจะทำให้โครงสร้างที่คิดไว้ผิดเพี้ยนไป ต่อจากนั้นเพิ่มโอเปอเรชัน (Operation) ในคลาสตามผลลัพธ์ของการสร้างไดนามิกโมเดลและฟังก์ชันนัลโมเดล โอเปอเรชันไม่สามารถกำหนดได้อย่างครบถ้วนจนกว่าจะเข้าใจไดนามิกโมเดลและฟังก์ชันโมเดลแล้ว วิธีที่ดีที่สุดคือเขียนความคิดต่างๆลงไปในกระดาษก่อน อย่าพึ่งพยายามที่จะจัดระเบียบความคิดเหล่านั้นมากจนเกินไปถึงแม้ว่าความคิดเหล่านั้นอาจจะซ้ำซ้อนหรือไม่คงเส้นคงวา เพื่อว่าจะได้ไม่ตกหล่นรายละเอียดที่สำคัญ แบบจำลองการวิเคราะห์ระบบเริ่มแรกนี้ดูเหมือนว่ายังมีข้อผิดที่จะต้องแก้ไขดังนั้นจะต้องมีการทำซ้ำ แบบจำลองทั้งหมดไม่จำเป็นต้องสร้างขึ้นมาเป็นแบบเดียวกันหมด

ขั้นตอนการสร้างออบเจคโมเดลมีดังนี้

- กำหนดออบเจค และคลาส
- จัดทำพจนานุกรมข้อมูล (data dictionary)
- กำหนดแอสโซซิเอชัน (รวมทั้ง aggregation) ระหว่างออบเจค
- กำหนดแอทริบิวท์ของออบเจคและลิงค์ (link)

4.3.1 กำหนดคลาส

ขั้นตอนแรกในการสร้างออบเจคโมเดลคือกำหนดคลาสที่อยู่ในขอบเขตของระบบงาน ออบเจคหมายถึงสิ่งของที่มีตัวตน เช่น บ้าน ลูกจ้าง และเครื่องจักร เป็นต้น ออบเจคยังหมายถึงแนวความคิด เช่น การจองที่นั่งโดยสารสายการบิน ตารางการจ่ายเงินเดือน เป็นต้น ทุกๆ คลาสต้องอยู่ภายในขอบเขตของระบบงาน หลีกเลี่ยงสิ่งที่เป็นโครงสร้างทางคอมพิวเตอร์ เช่นลิงค์ลิสท์ (link list) ซับรูทีน (subroutine) เป็นต้น ทุกๆ คลาสไม่ได้มีอยู่อย่างชัดเจนในประโยคปัญหา บางคลาสได้มาจากขอบเขตของระบบงานหรือจากความรู้โดยทั่วๆ ไป เขียนชื่อของคลาสทั้งหมดที่คิดได้โดยไม่ต้องเลือกมากจนเกินไป คลาสมักจะตรงกับคำนาม อย่าไปกังวลเกี่ยวกับการอินเฮริทันซ์หรือเกี่ยวกับคลาสในระดับสูงมากนัก ตัวอย่างเช่น การทำรายการสิ่งของและการตรวจสอบ

สำหรับระบบงานห้องสมุด กำหนดชนิดสิ่งของที่แตกต่างกัน เช่น หนังสือ วารสารหนังสือพิมพ์ เทปวีดีโอ เป็นต้น ต่อจากนั้นรวมกลุ่มสิ่งของตามลักษณะที่คล้ายหรือแตกต่างกัน

ตัดคลาสที่ไม่จำเป็นและไม่ถูกต้องออกโดยใช้กฎเกณฑ์ดังต่อไปนี้

- คลาสที่ซ้ำซ้อนกัน (redundant classes) ถ้าคลาสแทนสิ่งเดียวกันเลือกชื่อคลาสที่มีความหมายอธิบายถึงสิ่งนั้นได้ชัดเจนดีที่สุดไว้ ตัวอย่างเช่น คลาส Customer และ Passenger ถ้าหมายถึงผู้โดยสารเครื่องบินเหมือนกันชื่อคลาส Passenger ย่อมอธิบายได้ตรงกว่า แต่ถ้าหมายถึงคนที่ติดต่อกับสายการบิน ชื่อคลาส Person ย่อมเหมาะสมกว่า เพราะความหมายใช้คลุมไปถึงผู้โดยสารได้ด้วย

- คลาสที่ไม่เกี่ยวข้องกับขอบเขตของปัญหา หรือเกี่ยวข้องน้อยมากควรที่จะตัดคลาสนั้นออก ตัวอย่างเช่น ระบบจองบัตรละคร คลาส "คนเก็บบัตร" ไม่เกี่ยวข้อง แต่คลาส 'พนักงานโรงละคร' อาจจะมีความเกี่ยวข้อง

- คลาสที่คลุมเครือไม่ชัดเจนบางคลาสอาจจะกำหนดขอบเขตกว้างจนเกินไป ตัวอย่างเช่น คลาส "เนตเวิร์ค" เป็นคลาสที่กว้างเกินไป

- คลาสที่ควรจะเป็นแอทริบิวท์ ตัวอย่างเช่น ชื่อ อายุ ที่อยู่ โดยปกติเป็นแอทริบิวท์แต่ถ้ามีคุณสมบัติอิสระที่สำคัญควรจะกำหนดเป็นคลาสไม่ใช่แอทริบิวท์ ตัวอย่างเช่น ห้องทำงานเป็นคลาสในระบบการจัดห้อง

- ชื่อคลาสซึ่งอธิบายโอเปอเรชันของออบเจคและไม่ได้มีการกระทำภายในตัวเองดังนั้นไม่ใช่คลาส ตัวอย่างเช่น call เป็นผลจากการกระทำของผู้เรียกและเนตเวิร์คของโทรศัพท์ ถ้าเป็นระบบโทรศัพท์ธรรมดา Call เป็นเพียงส่วนหนึ่งของไดนามิกโมเดลเท่านั้นไม่ใช่คลาส แต่ถ้าโอเปอเรชันมีคุณลักษณะของตนเองควรกำหนดเป็นคลาส ตัวอย่างเช่น ระบบเก็บเงินค่าใช้โทรศัพท์ Call ควรจะเป็นคลาสและมีแอทริบิวท์เป็น วัน เวลา และจุดหมายปลายทาง เป็นต้น

- คลาสควรจะมีชื่อที่หมายถึงเนื้อแท้ของออบเจค ไม่ใช่บทบาทในแอสโซซิเอชัน ตัวอย่างเช่น Owner เป็นชื่อคลาสที่ไม่ดีในฐานข้อมูลของโรงงานผลิตรถ เพราะว่าถ้าจะเพิ่มคนขับหรือคนเช่ารถในภายหลังจะทำอย่างไร ควรจะใช้ชื่อว่า Person หรือ Customer ซึ่งจะมีความหมายรวมได้หลายบทบาทเช่น เจ้าของ คนขับ และผู้เช่า บางครั้งเอ็นทิตี้ (entity) หนึ่งตรงกับคลาสได้หลายคลาส ตัวอย่างเช่น Person กับ Employee อาจจะเป็นคลาสที่แตกต่างกันในบางพฤติกรรม หรืออาจจะเป็นคลาสที่ซ้ำซ้อนกันในพฤติกรรมอื่นๆ ถ้าเป็นมุมมองของฐานข้อมูลของลูกจ้างบริษัท คลาสทั้งสองคลาสนี้อาจจะเป็นคลาสเดียวกัน แต่สำหรับฐานข้อมูลเกี่ยวกับการเก็บภาษีของรัฐบาล คนสามารถทำงานได้มากกว่าหนึ่งงาน ดังนั้นจึงเป็นเรื่องที่สำคัญที่จะต้อง

กำหนดความแตกต่างของคลาส Person กับ Employee คนหนึ่งสามารถเป็นสมาชิกๆหนึ่งของ Employee หรือเป็นมากกว่าหนึ่งสมาชิก หรือไม่เป็นสมาชิกเลย

- คลาสที่เป็นโครงสร้างของการสร้างระบบงาน ควรจะตัดออกจากการวิเคราะห์ระบบ เพราะว่าสิ่งเหล่านั้นยังไม่ต้องการตอนนี้ แต่จะต้องการในตอนออกแบบระบบ ตัวอย่างเช่น ซีพียู ซับรูทีน อะกอริทึม ลิงค์ลิส ทรี (tree) เป็นต้น

4.3.2 กำหนดแอทริบิวท์

ต่อไปกำหนดแอทริบิวท์ แอทริบิวท์คือคุณสมบัติเฉพาะของแต่ละออบเจคเช่นชื่อ น้ำหนัก หรือสี แอทริบิวท์ไม่ควรจะเป็นออบเจค เราควรใช้แอสโซซิเอชันแสดงความสัมพันธ์ระหว่างออบเจคสองออบเจค โดยปกติแอทริบิวท์จะตรงกับคำนามที่ตามด้วยวลีแสดงความเป็นเจ้าของเช่น สีของรถหรือ ตำแหน่งของเคอร์เซอร์ (cursor) คำคุณศัพท์ (adjectives) มักจะใช้แทนค่าของแอทริบิวท์เช่น สีแดง(red) อยู่(on) หรือสิ้นสุด(expired) แอทริบิวท์แทบจะไม่มีอยู่ในประโยคปัญหาซึ่งแตกต่างกับคลาสและแอสโซซิเอชัน ดังนั้นจึงต้องค้นหาเอาเองจากขอบเขตของระบบงานและตามความเป็นจริง โชคดีที่นานๆ ครั้งแอทริบิวท์จะมีอิทธิพลเหนือโครงสร้างพื้นฐานของปัญหา อย่ากำหนดแอทริบิวท์มากจนเกินไป คิดเฉพาะแอทริบิวท์ที่อยู่ในระบบงานเท่านั้น อันดับแรกให้เลือกแอทริบิวท์ที่สำคัญที่สุด ส่วนของรายละเอียดเพิ่มเติมในภายหลังได้ ในระหว่างการวิเคราะห์หลีกเลี่ยงแอทริบิวท์ที่เกี่ยวกับการพัฒนา ชื่อของแอทริบิวท์ต้องมีความหมายที่ทำให้เข้าใจได้หรือกำหนดอย่างชัดเจน ควรจะละเว้นแอทริบิวท์ที่ได้มาจากแอทริบิวท์อื่น ตัวอย่างเช่น อายุได้มาจากวันเกิดและเวลาปัจจุบัน ดิไรว์แอทริบิวท์สามารถใช้กำหนดคุณสมบัติแบบนามธรรมของระบบงาน แต่ว่าต้องกำหนดให้แตกต่างกับแอทริบิวท์หลักอย่างชัดเจน ซึ่งแอทริบิวท์หลักจะกำหนดสถานะของออบเจค ดิไรว์แอทริบิวท์จะต้องไม่ดูเหมือนกับเป็นโอเปอเรชัน เช่น หาอายุ (get-age) ถึงแม้ว่ามันอาจจะมีการกระทำเป็นแบบนั้นก็ตาม กำหนดลิงค์แอทริบิวท์ด้วยเช่นกัน ลิงค์แอทริบิวท์คือคุณสมบัติของลิงค์ที่อยู่ระหว่างสองออบเจคไม่ใช่เป็นคุณสมบัติของตัวออบเจค ตัวอย่างเช่น แอสโซซิเอชันแบบ many-to-many ซึ่งอยู่ระหว่างผู้ถือพันธบัตรกับบริษัท มีลิงค์แอทริบิวท์คือจำนวนการใช้ร่วมกัน

คัดเลือกแอทริบิวท์ที่ไม่จำเป็นและไม่ถูกต้องออกโดยใช้กฎเกณฑ์ดังต่อไปนี้

- ถ้าสิ่งที่มีอยู่แล้วอย่างเป็นอิสระมีความสำคัญมากกว่าค่าของมัน ดังนั้นสิ่งนั้นคือออบเจค ตัวอย่างเช่น หัวหน้าคือออบเจค และเงินเดือนคือแอทริบิวท์ การแยกความแตกต่างนี้มักจะขึ้นกับระบบงาน ตัวอย่างเช่น ระบบการส่งทางไปรษณีย์ เมืองคือแอทริบิวท์ ขณะที่ในระบบ

สำมะโนประชากร เมืองควรจะเป็นออบเจค สิ่งใดที่มีลักษณะเป็นของตนเองภายในระบบงานที่กำหนดให้สิ่งนั้นคือออบเจค

-ถ้าค่าของแอทริบิวท์ขึ้นกับข้อความที่ระบุเฉพาะดังนั้นแอทริบิวท์นี้เป็นควอลลิไฟเออร์ตัวอย่างเช่น หมายเลขพนักงานไม่ได้มีเพียงค่าเดียวสำหรับคนที่ทำงานสองงาน หมายเลขพนักงานจะทำให้แอสโซซิเอชันใน "บริษัทจ้างคนทำงาน" ชัดเจนขึ้น

- ชื่อมักจะถูกกำหนดเป็นควอลลิไฟเออร์มากกว่าเป็นแอทริบิวท์ของออบเจค ทดสอบดูว่าชื่อนั้นใช้ระบุออบเจคที่อยู่ในกลุ่มได้ไหม ออบเจคภายในกลุ่มสามารถมีชื่อได้มากกว่าหนึ่งชื่อไหม ถ้าตอบว่าใช่ชื่อนั้นทำให้แอสโซซิเอชันมีความชัดเจนดีขึ้น ถ้าปรากฏว่าชื่อนี้มีค่าไม่ซ้ำกันแสดงว่ามีคลาสที่ถูกควอลลิไฟขาดหายไปตัวอย่างเช่น ชื่อของแผนกจะไม่ซ้ำกันในบริษัทหนึ่ง แต่ในบางโปรแกรมอาจจะต้องการทำกับหลายบริษัทซึ่งมีชื่อของแผนกที่ซ้ำกัน ชื่อจะเป็นแอทริบิวท์ของออบเจคก็ต่อเมื่อชื่อนั้นไม่ได้ขึ้นกับเนื้อหาในข้อความ และโดยเฉพาะเมื่อชื่อนั้นมีค่าซ้ำกันได้ ชื่อของคนต่างกับชื่อของบริษัทซึ่งชื่อคนอาจจะมีค่าซ้ำกันได้ เพราะฉะนั้นจึงเป็นแอทริบิวท์ของออบเจค

- ภาษาเชิงวัตถุนั้นได้รวมความคิดเกี่ยวกับค่าที่ใช้ระบุตัวตนของออบเจคหนึ่งๆสำหรับใช้อ้างถึงออบเจคนั้นได้อย่างไม่คลุมเครือเข้าไว้ในภาษาด้วย ในออบเจคโมเดลไม่ต้องแสดงค่าที่ใช้ระบุตัวตนของออบเจคเหล่านี้ เหมือนกับว่ามีค่าเหล่านี้ในออบเจคโมเดลอยู่แล้ว แสดงเฉพาะค่าของแอทริบิวท์ที่อยู่ในขอบเขตของระบบงานเท่านั้น ตัวอย่างเช่น หมายเลข serial number ของพัสดุใช้อ้างถึงพัสดุแต่ละชิ้นส่วน

- ถ้าคุณสมบัติขึ้นกับการแทนค่าของลิงค์ ดังนั้นคุณสมบัตินั้นคือแอทริบิวท์ของลิงค์และไม่เกี่ยวข้องกับออบเจค แอทริบิวท์ของลิงค์โดยปกติปรากฏชัดบนแอสโซซิเอชันแบบ many-to-many ซึ่งแอทริบิวท์ของลิงค์ไม่สามารถจะติดเข้ากับคลาสทั้งสองเพราะมัลติพลิซิตี้ของคลาสทั้งสอง แอทริบิวท์ของลิงค์บนแอสโซซิเอชันแบบ many-to-one นั้นดีกว่าเพราะสามารถติดเข้ากับออบเจคด้านที่เป็น many ได้โดยข่าวสารไม่หาย แอทริบิวท์บนแอสโซซิเอชันแบบ one-to-one ก็สามารถทำได้เช่นเดียวกัน

- ถ้าแอทริบิวท์อธิบายสถานะภายในของออบเจคซึ่งภายนอกออบเจคจะมองไม่เห็นดังนั้นตัดมันออกจากการวิเคราะห์

- ละเว้นแอทริบิวท์รองๆลงมาซึ่งไม่น่าจะเป็นไปได้ว่าจะมีอิทธิพลเหนือโอเปอเรชันส่วนมาก

- แอทริบิวท์ที่ดูเหมือนว่าแตกต่างกับแอทริบิวท์อื่น และไม่เกี่ยวพันกับแอทริบิวท์อื่นๆ อาจจะชี้คลาสซึ่งควรจะแตกออกเป็นสองคลาสที่แตกต่างกัน คลาสควรจะเป็นแบบธรรมดาและมีการติดต่อกัน

ขั้นตอนต่อไปคือจัดคลาสโดยการใช้อินเฮริทันซ์เพื่อใช้โครงสร้างแบบสามัญร่วมกัน อินเฮริทันซ์สามารถเพิ่มเข้ามาได้สองทางด้วยกัน คือโดยกำหนดลักษณะแบบธรรมดาทั่วๆ ไปของคลาสที่มีอยู่แล้วใส่เข้าไปในซูปเปอร์คลาส(superclass) เรียกว่าวิธีทำจากข้างล่างขึ้นมา(bottom up)หรือโดยการขัดเกลาคลาสที่มีอยู่แล้วใส่เข้าไปในซับคลาสพิเศษเรียกว่าวิธีทำจากข้างบนลงไป (top down)

เราสามารถจะค้นพบอินเฮริทันซ์จากวิธีทำจากข้างล่างขึ้นมาโดยการหาคลาสที่มีแอทริบิวท์ แอสโซซิเอชัน หรือโอเปอเรชันคล้ายๆ กันกำหนดเป็นซูปเปอร์คลาสที่ใช้ลักษณะแบบสามัญร่วมกันสำหรับแต่ละเจนเนอรัลไลเซชัน(generalization) ตัวอย่างเช่น พัสดุสิ้นเปลือง พัสดุซ่อมหมุนเวียน สามารถที่จะสร้างแบบรวมๆ ได้คือ "พัสดุ" บางแอทริบิวท์หรือคลาสที่เสมอกันอาจจะถูกกำหนดใหม่เพียงเล็กน้อยเท่านั้นที่จะทำให้เหมาะสมกันโดยถูกต้อง แต่อย่าพยายามที่จะทำให้เข้ากันได้มากจนเกินไปนัก เพราะอาจจะได้เจนเนอรัลไลเซชันที่ผิด บางเจนเนอรัลไลเซชันซึ่งมีอยู่จริงอาจจะพบได้ในที่กำลังจัดกลุ่ม ใช้แนวความคิดที่มีอยู่แล้วนี้เมื่อไรก็ตามที่เป็นไปได้ ความสมดุลจะเสนอแนะคลาสที่ขาดหายไปจากเจนเนอรัลไลเซชันนั้น

วิธีทำจากข้างบนลงไป สเพชชัลไลเซชัน (specialization) มักจะปรากฏขึ้นมาจากขอบเขตของระบบงาน ให้มองหาวลีคำนามประกอบด้วยคำคุณศัพท์หลายชนิดบนชื่อคลาส เช่น fluorescent lamp,incandescent lamp ให้หลีกเลี่ยงการขัดเกลาที่มากจนเกินไป ถ้าสเพชชัลไลเซชันที่เสนอขึ้นมานี้ขัดแย้งกับคลาสที่มีอยู่แล้ว คลาสที่มีอยู่แล้วอาจจะถูกกำหนดไว้ไม่ถูก กรณีรองๆลงมาในขอบเขตของระบบงานทั้งหมดคือแหล่งกำเนิดของสเพชชัลไลเซชัน บ่อยครั้งที่เขียนเฉพาะเซตของกรณีรองๆ ที่มีอยู่ก็เป็นการเพียงพอแล้วโดยไม่ต้องเขียนที่เป็นจริงทั้งหมด ตัวอย่างเช่น Bank account สามารถขัดเกลาเป็น Checking account และ Savings account ในขณะที่ในระบบงานธนาคารบางระบบความแตกต่างนี้ไม่มีอิทธิพลเหนือพฤติกรรมของระบบ เพียงแค่เพิ่มแอทริบิวท์ธรรมดาคือ "account type" ลงใน account เท่านั้นก็เพียงพอ

มัลติเพิลอินเฮริทันซ์(multiple inheritance) อาจจะนำมาใช้เพื่อเพิ่มการใช้ร่วมกันแต่ต้องเฉพาะเมื่อจำเป็นเท่านั้น เพราะว่ามันเพิ่มทั้งแนวความคิดและความซับซ้อนในการสร้างระบบ

เมื่อชื่อของแอสโซซิเอชันที่เหมือนกันปรากฏมากกว่าหนึ่งครั้งด้วยความหมายแบบเดียวกัน ดังนั้นพยายามกำหนดอย่างกว้างๆสำหรับคลาสที่รวมกัน ตัวอย่างเช่น Transaction ถูกส่ง

เข้ามาทั้งทาง Cashier และ ATM กำหนดเจนเนอรัลไลเซชันคือ Entry station สำหรับ Cashier station และ ATM ในบางครั้งคลาสไม่มีอะไรที่คล้ายๆ กันเลยนอกจากแอสโซซิเอชัน แต่ก็มีบ่อยครั้งที่พบลักษณะโดยทั่วไปที่เคยถูกมองข้าม

แอทริบิวท์และแอสโซซิเอชันต้องกำหนดกับคลาสที่ระบุเฉพาะ ซึ่งอยู่ในแผนผังของคลาส(class hierarchy) แต่ละแอทริบิวท์และแอสซิเอชันต้องถูกกำหนดกับคลาสแบบทั่วไปส่วนมากที่เหมาะสม การทำทุกอย่างให้ถูกต้องอาจจะต้องการการแก้ไขบางอย่างให้เหมาะสม ความสมดุลอาจจะเสนอแนะแอทริบิวท์เพิ่มขึ้นเพื่อบอกความแตกต่างระหว่างซับคลาสให้เห็นชัดเจนขึ้น

ให้ลองไล่ตามเส้นทางต่างๆ ตลอดทั้งออบเจคโมเดลเพื่อมองดูว่าเส้นทางนั้นให้ผลลัพธ์ที่สมเหตุสมผลหรือไม่ ในที่ๆ ที่คาดว่ามีเพียงแค่ค่าเดียวจะมีเส้นทางที่ให้ผลลัพธ์เพียงค่าเดียวหรือไม่ สำหรับมัลติพลิซิตี้ (multiplicity) แบบ "many" มีหนทางที่จะได้ค่าเดียวเมื่อต้องการหรือไม่ คิดถึงคำถามที่อยากจะถามมีคำถามที่มีประโยชน์ซึ่งตอบไม่ได้หรือไม่ สิ่งเหล่านี้จะบ่งชี้ถึงข่าวสารที่ขาดหายไป ถ้ามีบางสิ่งซึ่งดูธรรมดาในความเป็นจริง แต่กลับซับซ้อนในแบบจำลองแสดงว่าอาจจะมีบางสิ่งที่ขาดหายไป (แต่ต้องมั่นใจว่ามันไม่ได้ซับซ้อนจริงๆ)

มันยากมากที่ออบเจคโมเดลจะถูกต้องหลังการทำเพียงครั้งเดียว การพัฒนาซอฟท์แวร์ทั้งหมดคือหนึ่งครั้งของการทำซ้ำ ส่วนที่แตกต่างกันของแบบจำลองมักจะอยู่ที่ขั้นของความสำเร็จซึ่งแตกต่างกันด้วย ถ้าพบความบกพร่องให้ย้อนกลับไปที่ขั้นก่อนหน้านั้นถ้ามันจำเป็นต้องแก้ไขการขัดเกลาบางอย่างจะสามารถทำได้ก็ต่อเมื่อทำไดนามิกโมเดลและฟังก์ชั่นนัลโมเดลเสร็จเรียบร้อยแล้ว

สัญญาณที่บอกว่ามีออบเจคขาดหายไป

- ความสมดุลในแอสโซซิเอชันและเจนเนอรัลไลเซชัน: ให้เพิ่มคลาสใหม่โดยใช้ความคล้ายคลึงกัน

- กระจายแอทริบิวท์และโอเปอเรชันของคลาส: ให้แตกคลาสออก ดังนั้นแต่ละส่วนจะมีการติดต่อกัน

- คุณสมบัติร่วมกันของคลาสหารำบาก: คลาสหนึ่งอาจแสดงสองบทบาท แตกคลาสนั้นออก

- โอเปอเรชันที่ไม่มีคลาสเป้าหมายที่ดี: ให้เพิ่มคลาสเป้าหมายที่ขาดไป

- แอสโซซิเอชันที่ซ้ำกันทั้งชื่อและความประสงค์: ให้สร้างซุปเปอร์คลาสขึ้น

- บทบาทของคลาส: อาจจะต้องเป็นคลาสที่แยกกัน ซึ่งมักจะหมายถึงการเปลี่ยนแอสโซซิเอชันไปเป็นคลาส ตัวอย่างเช่น คนสามารถที่จะทำงานได้หลายบริษัท ดังนั้น Employee คือ

คลาสของคนทำงานบริษัทซึ่งระบุว่าเป็นบริษัทอะไร ซึ่งเป็นคลาสที่เพิ่มขึ้นมาจากคลาส Person และ Company

สัญญาณที่บอกว่าคลาสนั้นไม่จำเป็น

- แอทริบิวท์ โอเปอเรชัน และ แอสโซซิเอชัน บนคลาสที่ขาดไป: ให้ตั้งคำถามว่าทำไมถึงต้องการสิ่งนี้

สัญญาณที่บอกว่าแอสโซซิเอชันนั้นขาดไป

- เส้นทางสำหรับโอเปอเรชันขาดไป: ให้เพิ่มแอสโซซิเอชันเพื่อให้สามารถตอบคำถามได้

สัญญาณที่บอกว่าแอสโซซิเอชันนั้นไม่จำเป็น

- ข่าวสารในแอสโซซิเอชันที่ซ้ำซ้อน: ให้ตัดแอสโซซิเอชันที่ไม่ได้เพิ่มข่าวสารใหม่หรือเปลี่ยนไปเป็นดิไรว์แทน

- แอสโซซิเอชันที่ไม่มีโอเปอเรชัน: ถ้าไม่มีโอเปอเรชันใช้เส้นทางนั้นอาจจะไม่ต้องการข่าวสารนั้น ซึ่งการทดสอบนี้ต้องคอยจนกระทั่งกำหนดโอเปอเรชันเสร็จแล้ว

สัญญาณที่บอกว่ามีแอสโซซิเอชันวางไม่ถูกที่

- ชื่อของบทบาทซึ่งกว้างหรือแคบจนเกินไป: ให้เคลื่อนย้ายแอสโซซิเอชันขึ้นหรือลงในแผนผังของคลาสเพื่อหาตำแหน่งที่เหมาะสม

สัญญาณที่บอกว่ามีแอทริบิวท์วางไม่ถูกที่

- ต้องการเข้าถึงออบเจคโดยผ่านทางค่าแอทริบิวท์ค่าหนึ่งของออบเจคนั้น: ให้กำหนดควอลลิไฟแอสโซซิเอชัน (qualified association) ขึ้น

ในวิธีการปฏิบัติ การสร้างแบบจำลองไม่ได้ต้องทำตามลำดับที่กล่าวมาข้างต้นอย่างเข้มงวด เราสามารถที่จะรวมหลายขั้นตอนเข้าด้วยกันได้เมื่อเรามีประสบสบการณ์ ตัวอย่างเช่น เราสามารถกำหนดคลาส ตัดคลาสที่ไม่ถูกต้องออกโดยไม่ต้องเขียนคลาสนั้นลงไป และใส่คลาสเหล่านั้นลงในแผนผังออบเจคพร้อมกับแอสโซซิเอชัน เราสามารถเลือกบางส่วนของแบบจำลองและพัฒนาในรายละเอียดบางอย่าง ในขณะที่ส่วนอื่นๆ ยังเป็นร่างอยู่ ลำดับขั้นตอนสามารถจะสลับกันได้ถ้าเหมาะสม แต่ถ้าพึ่งจะเรียนรู้ออบเจคโมเดล ขอแนะนำให้ทำตามลำดับขั้นตอน

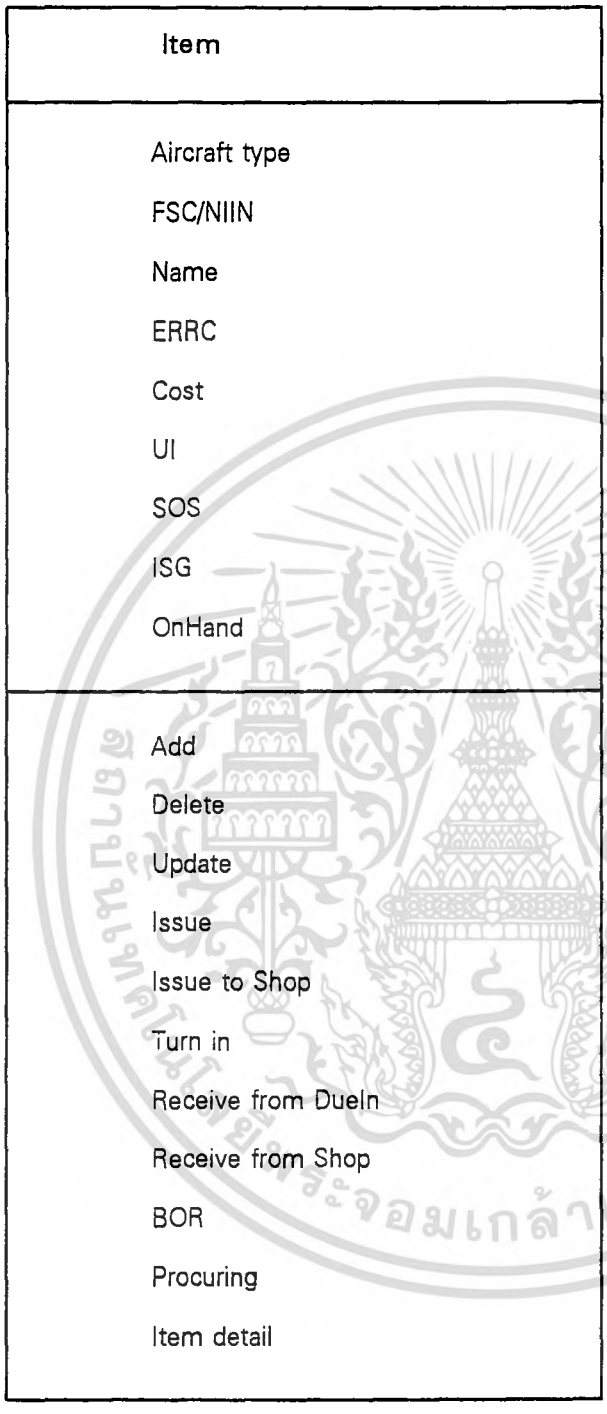
ขั้นตอนสุดท้ายของออบเจคโมเดลคือจัดกลุ่มคลาสลงในหน้ากระดาษและในโมดูล (module) แผนผังอาจจะแบ่งออกเป็นหน้ากระดาษที่มีขนาดและรูปแบบเดียวกันเพื่อความสะดวกใน การวาด การเขียน และการดู คลาสที่ยึดติดกันต้องรวมกลุ่มเข้าด้วยกัน แต่หน้า

กระดาษมีขนาดจำกัด ดังนั้นต้องมีการตัดแบ่งเป็นครั้งคราวตามแต่จะกำหนดเอาเอง โมดูลคือกลุ่มของคลาส (หนึ่งหน้ากระดาษหรือมากกว่า) ซึ่งเป็นบางส่วนของแบบจำลองทั้งหมด ตัวอย่างเช่น แบบจำลองของระบบปฏิบัติการคอมพิวเตอร์อาจประกอบด้วยโมดูลของควบคุมการทำงาน ควบคุมอุปกรณ์ บำรุงรักษาแฟ้มข้อมูล และจัดหน่วยความจำ ขนาดของโมดูลเปลี่ยนแปลงได้

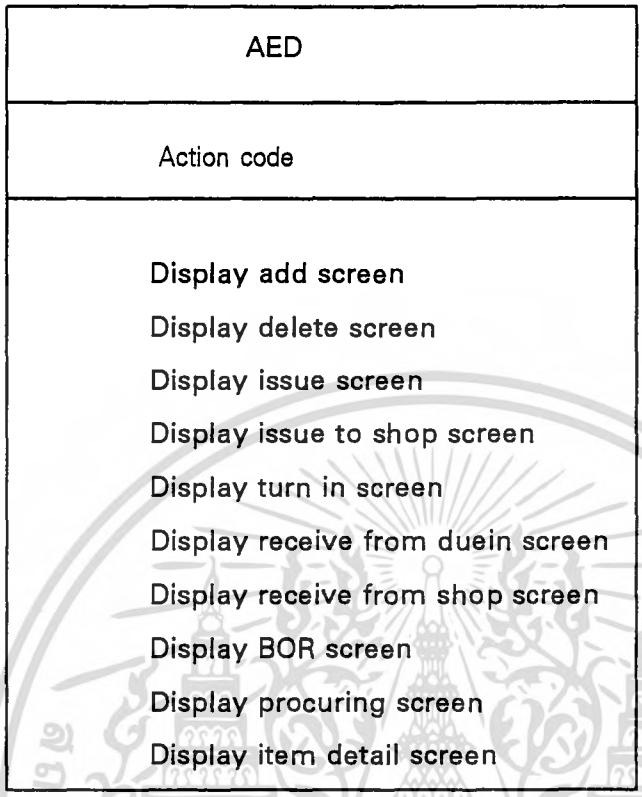
แต่ละแอสโซซิเอชันควรจะอยู่ในหน้ากระดาษเดียวกัน แต่บางคลาสอาจจะปรากฏมากกว่าหนึ่งครั้งเพื่อใช้เป็นจุดต่อหน้ากระดาษ หาจุดตัดในระหว่างคลาส คลาสซึ่งเป็นจุดต่อจุดเดียวในระหว่างสองหน้ากระดาษหรือโมดูล หรือคือจุดที่แยกออบเจคเนตเวิร์ค คลาสนั้นเป็นเหมือนสะพานระหว่างสองหน้ากระดาษหรือโมดูล ตัวอย่างเช่นในระบบจัดการแฟ้มข้อมูล แฟ้มข้อมูลคือจุดตัดระหว่างโครงสร้างของไดเรคทอรี และเนื้อเรื่องของแฟ้มข้อมูล ถ้าหาจุดตัดเพียงจุดเดียวไม่ได้ก็พยายามให้มีจุดเชื่อมต่อน้อยที่สุด พยายามเลือกใช้โมดูลเพื่อลดจำนวนของการข้ามหน้ากระดาษในแผนผังออบเจค แผนผังออบเจคส่วนมากสามารถเขียนเป็นกราฟได้โดยที่ไม่มีเส้นตัดกัน

นำโมดูลที่ออกแบบก่อนหน้านั้นกลับมาใช้อีกถ้าเป็นไปได้ แต่อย่าพยายามที่จะทำให้มันเป็นเช่นนั้น การนำกลับมาใช้ใหม่(reuse) เป็นสิ่งที่ง่ายที่สุดเมื่อส่วนหนึ่งของปัญหาเข้ากันได้กับปัญหาก่อนหน้านี้ ถ้าปัญหาใหม่คล้ายๆ กับปัญหาก่อนหน้านี้ การออกแบบเริ่มแรกจะต้องขยายกว้างออกไปเพื่อให้ครอบคลุมได้ทั้งสองปัญหา

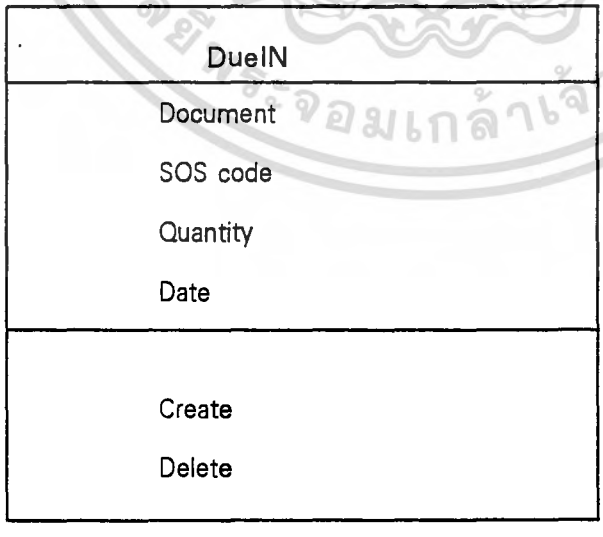
รายละเอียดของคลาสต่างๆ ที่กำหนดขึ้นได้ในระบบงานบริหารพัสดุสายช่างอากาศมีดังนี้คือ



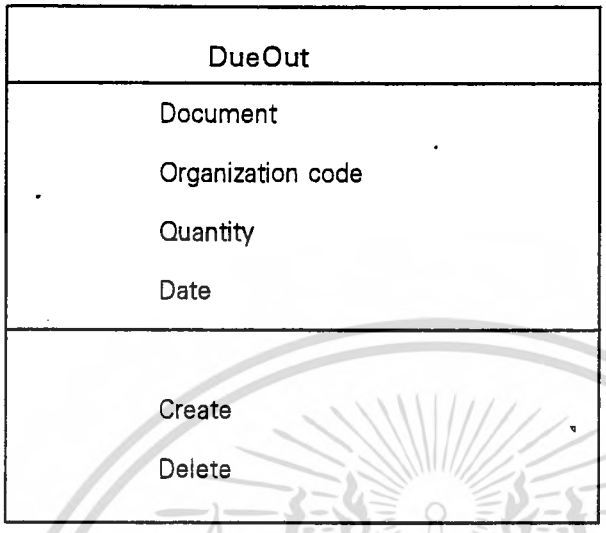
คลาส Item เป็นคลาสของพัสดุ



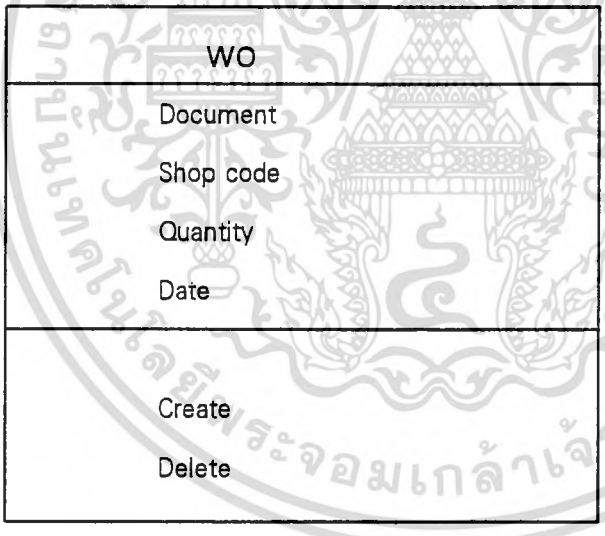
คลาส AED เป็นคลาสที่ใช้ในการรับข้อความติดต่อจากผู้ใช้โดยตรงแล้วส่งผ่านต่อไปยังคลาสต่างๆ ภายในระบบงาน หรือเป็นคลาสของระบบงานนั้นเอง



คลาส DueIn เป็นคลาสของเอกสารรอรับการจัดหา



คลาส DueOut เป็นคลาสของเอกสารการค้างจ่าย



คลาส WO เป็นคลาสของเอกสารรอรับงานซ่อมพัสดุ

4.3.2 การจัดทำพจนานุกรมข้อมูล

เนื่องจากว่าคำแต่ละคำมีความหมายได้หลายอย่างดังนั้นควรทำพจนานุกรมข้อมูลสำหรับแบบจำลองทั้งหมด เขียนคำอธิบายแต่ละคลาสอย่างครบถ้วนอธิบายขอบเขตของคลาสภายในปัญหาขณะนั้น

พจนานุกรมข้อมูลของระบบงานบริหารพัสดุมีดังนี้

ชื่อข้อมูล	ความหมาย
Aircraft type	รหัสชนิดของเครื่อง
FSC/NIIN	รหัสหมายเลขของพัสดุ
Name	ชื่อของพัสดุ
ERRC	รหัสขีดความสามารถในการซ่อมได้ของพัสดุ
Cost	ราคาต่อหน่วยของพัสดุ
UI	รหัสหน่วยนับของพัสดุ
SOS	รหัสของแหล่งที่จัดหาพัสดุ
ISG	รหัสของหมายเลขพัสดุที่ใช้แทนกันได้
OnHand	จำนวนพัสดุดี
Repair	จำนวนพัสดุชำรุด
Action code	รหัสระบุประเภทของการทำงาน (รายละเอียดอยู่ในรูปที่ 5.4)
Document	หมายเลขของเอกสาร แบ่งออกเป็นของ คลังใหญ่ หน่วยผู้ใช้ เอกสารการซ่อมพัสดุ เอกสารการจัดหาพัสดุ
SOS code	รหัสของแหล่งที่จัดหาพัสดุ
Shop code	รหัสของหน่วยที่ซ่อมพัสดุ
Organization code	รหัสของหน่วยผู้ใช้
Quantity	จำนวนของพัสดุที่จ่าย ส่งซ่อม หรือจัดหา
Date	วันที่จ่าย ส่งซ่อม หรือจัดหาพัสดุ

4.3.3 กำหนดแอสโซซิเอชัน

ต่อไปกำหนดแอสโซซิเอชันระหว่างคลาส ความสัมพันธ์ระหว่างคลาสสองคลาสหรือมากกว่าสองคลาสคือแอสโซซิเอชัน คลาสหนึ่งอ้างถึงคลาสอื่นโดยใช้แอสโซซิเอชัน จะต้องไม่อ้างถึงคลาสด้วยแอทริบิวท์ตัวอย่างเช่น คลาส Person ต้องไม่มีแอทริบิวท์เป็น Employer แต่ใช้แอสโซซิเอชันแทนเช่น แอสโซซิเอชัน Work-for ใช้อ้างถึงคลาส Company จาก Person แอสโซซิเอชัน

แสดงความสัมพันธ์ระหว่างคลาสในระดับเดียวกัน แอสโซซิเอชันถูกพัฒนาขึ้นมาได้หลายทางด้วยกัน แต่ยังไม่ควรเลือกว่าจะทำวิธีใดในแบบจำลองการวิเคราะห์เพื่อให้การออกแบบมีอิสระ แอสโซซิเอชันมักจะตรงคำกิริยาหรือวลีกิริยารวมทั้งสถานที่ (next to, part of,contained in) การกระตุ้น (drives) การติดต่อสื่อสาร (talk to) การเป็นเจ้าของ (has, part of) หรือการสอดคล้องกับเงื่อนไขใดเงื่อนไขหนึ่ง (work for, married to, manages) เลือกคำกิริยาเหล่านี้จากประโยคปัญหาเป็นแอสโซซิเอชันแล้วเขียนลงในกระดาษ ยังไม่ต้องรีบขัดเกลาอะไร อย่าพึ่งเสียเวลาที่จะพยายามแยกความแตกต่างระหว่างแอสโซซิเอชันกับอะกรีเกชัน (aggregation) อะกรีเกชันก็คือแอสโซซิเอชันพิเศษแบบหนึ่ง

ตัดแอสโซซิเอชันที่ไม่จำเป็นและไม่ถูกต้องออกโดยใช้กฎเกณฑ์ดังต่อไปนี้

- แอสโซซิเอชันที่อยู่ระหว่างคลาสที่ถูกตัดออก ถ้าคลาสใดคลาสหนึ่งในแอสโซซิเอชันถูกตัดออก ดังนั้นแอสโซซิเอชันต้องถูกตัดออกไปด้วย หรือต้องเปลี่ยนไปเป็นของคลาสอื่น

- ตัดแอสโซซิเอชันที่ไม่ถูกต้องในขอบเขตของปัญหาหรือเกี่ยวข้องกับโครงสร้างของการพัฒนา ตัวอย่างเช่น system handles concurrent access

- ตัดแอสโซซิเอชันที่อธิบายถึงการกระทำหรือเหตุการณ์ เพราะว่าแอสโซซิเอชันควรอธิบายถึงคุณสมบัติที่เป็นโครงสร้างของระบบงาน ตัวอย่างเช่น แผนกควบคุมรับเอกสารเบิกพัสดุ ซึ่งอธิบายถึงส่วนหนึ่งของการโต้ตอบกันระหว่างแผนกควบคุมกับหน่วยผู้ใช้ไม่ได้เป็นความสัมพันธ์ที่ถาวรระหว่างแผนกควบคุมกับเอกสารเบิกพัสดุแต่อย่างใด

- แอสโซซิเอชันที่เป็นชนิดเทอร์นารี(ternary)หรืออยู่ในระหว่างคลาสตั้งแต่สามคลาสขึ้นไปสามารถแตกออกเป็นชนิดไบนารีแอสโซซิเอชัน (binary) หรือควอลลิไฟแอสโซซิเอชัน(qualified association) ได้ก็ควรจะทำ ยกเว้นว่าไม่สามารถจะทำได้จริงๆ เพราะว่าข่าวสารบางส่วนจะสูญหายไป เช่น Professor teaches course in room ถ้าในเทอร์นารีแอสโซซิเอชันไม่มีลักษณะอะไรที่เป็นของตนเองดังนั้นสามารถทำเป็นแอทริบิวท์ของลิงค์ (link attribute) บนไบนารีแอสโซซิเอชันได้เช่น Company pays salary to person สามารถทำเป็น Company employs personมีค่า salary เป็นแอทริบิวท์ของลิงค์ Company-Person

- ตัดแอสโซซิเอชันที่ได้มาจากแอสโซซิเอชันอื่นๆ (derived association) เพราะว่าซ้ำซ้อนกัน ตัวอย่างเช่น Grandparent of นี้ตัดออกได้ เพราะว่าสามารถถูกกำหนดเป็นคู่ของ Parent of ได้ และถ้าเป็นแอสโซซิเอชันที่กำหนดมาจากเงื่อนไขของแอทริบิวท์ในออบเจค ตัดออกได้เช่นกัน เช่น younger than แทนเงื่อนไขวันเกิดของบุคคล (Person) สองคน ไม่ได้มีข่าวสารใหม่เพิ่ม

ขึ้น คลาส แอทริบิวท์ และแอสโซซิเอชันในออบเจคโมเดลควรจะแทนค่าของข่าวสารที่เป็นอิสระต่อกันมากที่สุดเท่าที่เป็นไปได้ เส้นทางระหว่างคลาสที่มีมากกว่าหนึ่ง (multiple paths) มักจะเป็นดิไรว์แอสโซซิเอชัน (derived association) ซึ่งเป็นส่วนประกอบของแอสโซซิเอชันเริ่มแรก (primitive association) แต่ต้องระวังเพราะว่าไม่ใช่ทุกแอสโซซิเอชันที่มีรูปแบบเป็นมัลติเพิลพาธแล้วจะกำหนดซ้ำซ้อนเสมอไป บางครั้งแอสโซซิเอชันสามารถมาจากสองแอสโซซิเอชันหรือมากกว่าสองได้ และทำมัลติพลิซิตี้ (multiplicity) ไม่ได้ให้เก็บแอสโซซิเอชันพิเศษนั้นไว้ ถ้าข้อบังคับของมัลติพลิซิตี้นั้นสำคัญ ตัวอย่างเช่น บริษัทจ้างคนทำงานหลายคนและเป็นเจ้าของคอมพิวเตอร์หลายเครื่อง ลูกจ้างบางคนไม่ถูกกำหนดให้ใช้คอมพิวเตอร์เลย แต่บางคนถูกกำหนดให้ใช้คอมพิวเตอร์ได้มากกว่าหนึ่งเครื่องคอมพิวเตอร์บางเครื่องมีคนใช้หนึ่งคนหรือไม่มีใครใช้เลยซึ่งแอสโซซิเอชัน Employs และ Owns ไม่สามารถอ้างถึงมัลติพลิซิตีของแอสโซซิเอชัน Assigned-to ได้ ถึงแม้ว่าดิไรว์แอสโซซิเอชันจะไม่ได้เพิ่มข่าวสารใหม่ แต่มีประโยชน์ในความเป็นจริงและในการออกแบบ ตัวอย่างเช่น ความสัมพันธ์กันในเครือญาติ เช่น ลุง แม่ยาย และหลาน อธิบายถึงความสัมพันธ์ของครอบครัวซึ่งมีความสำคัญในสังคมของเรา ในออบเจคโมเดลใช้สัญญลักษณ์เส้นประระบุดิไรว์แอสโซซิเอชันเพื่อให้แตกต่างกับแอสโซซิเอชันหลัก ต่อจากนั้นกำหนดแอสโซซิเอชันขั้นสูงดังต่อไปนี้

- การกำหนดชื่อของแอสโซซิเอชันนั้นเป็นเรื่องที่สำคัญ ไม่ต้องบอกว่าเป็นอย่างไรหรือทำไม บอกว่าคืออะไร ควรที่จะเลือกอย่างระมัดระวังและให้มีความหมายที่สามารถเข้าใจได้เช่น Bank computer maintains account เป็นประโยคของการกระทำ ประโยคของวลีกิริยาคือ Bank holds account

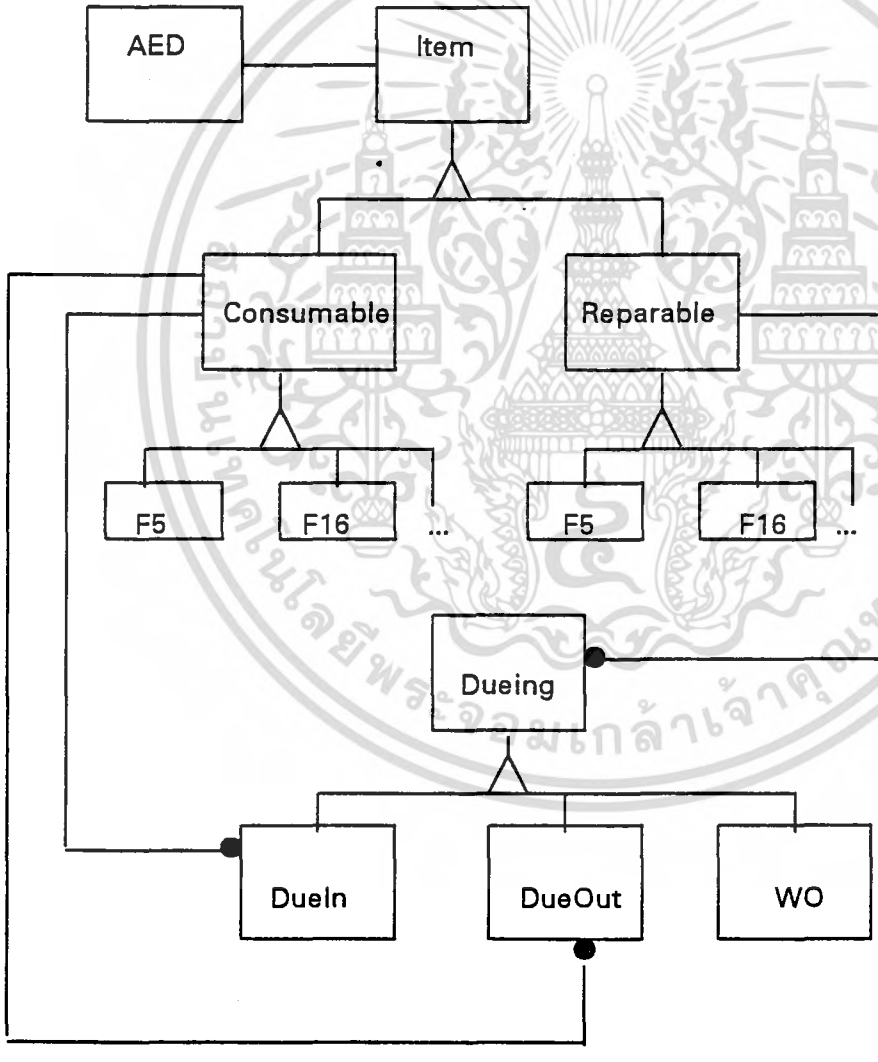
- เพิ่มชื่อของบทบาทในที่ๆ เหมาะสม ชื่อของบทบาทอธิบายถึงบทบาทของคลาสในแอสโซซิเอชันจากมุมมองของอีกคลาสหนึ่ง ตัวอย่างเช่นแอสโซซิเอชัน Work-for คลาส Company มีบทบาทเป็น employer และคลาส Person มีบทบาทเป็น employee แต่ถ้ามีเพียงแอสโซซิเอชันเดียวระหว่างคลาสคู่หนึ่งและชื่อของคลาสอธิบายบทบาทอย่างชัดเจนดีแล้วไม่ต้องมีชื่อของบทบาทเช่น Central computer communicates with ATM แอสโซซิเอชันระหว่างสองสมาชิกในคลาสเดียวกัน (reflexive association) ต้องมีชื่อบทบาท เพื่อแยกความแตกต่าง เช่น Person manages person ต้องมีบทบาท boss และ worker

- ควอลลิไฟแอสโซซิเอชัน(qualified association) โดยปกติชื่อจะบอกได้ว่าออบเจคนั้นๆ คืออะไร แต่ชื่อส่วนใหญ่แล้วจะซ้ำกัน เนื้อหารวมกับชื่อจะระบุให้ออบเจคนั้นไม่ซ้ำกัน ตัวอย่าง

เช่นชื่อของบริษัทในเมืองต้องไม่ซ้ำกัน แต่อาจจะซ้ำกับในเมืองอื่น ชื่อของบริษัทควอลลิไฟแอสโซซิเอชัน State charters company ชื่อของเมืองและบริษัทจะทำให้บริษัทไม่ซ้ำกัน ค่าควอลลิไฟจะบอกความแตกต่างของออบเจคด้านที่เป็นซนิด many ของแอสโซซิเอชัน ตัวอย่างเช่น หมายเลขเอกสารจะทำให้สามารถแยกเอกสารไปยังแผนกที่ถูกต้อง

- กำหนดมัลติพลิซิตี้ แต่อย่าพยายามที่จะทำให้มันถูกต้องมากจนเกินไปเพราะว่ามัลติพลิซิตี้มักจะเปลี่ยนแปลงในเวลาวิเคราะห์เสมอ
- เพิ่มแอสโซซิเอชันที่ขาดไป

รายละเอียดออบเจคโมเดลของระบบงานบริหารพัสดุสายช่างอากาศแสดงอยู่ในภาพที่ 34



ภาพที่ 34 ออบเจคโมเดล

4.4 ไดนามิกโมเดลลิ่ง (Dynamic modelling)

ไดนามิกโมเดลแสดงพฤติกรรม (behavior) ที่ขึ้นกับเวลาของระบบและออบเจคในแบบจำลอง เริ่มต้นการวิเคราะห์ไดนามิกโดยการหาอีเวนท์ (event) สิ่งเร้าและการตอบสนองที่มองเห็นได้จากภายนอกระบบ สรุปลำดับของอีเวนท์สำหรับแต่ละออบเจคลงในสเตทไดอะแกรม (state diagram) การทำอะกอริทึม (algorithm) สำเร็จไม่ใช่สิ่งที่การวิเคราะห์ต้องการทำถ้าไม่มีเหตุการณ์ที่มองเห็นได้จากภายนอกระบบ อะกอริทึมเป็นเพียงส่วนหนึ่งของการสร้างระบบเท่านั้น

ไดนามิกโมเดลไม่มีความสำคัญสำหรับงานซึ่งเกี่ยวกับการเก็บข้อมูลแท้ๆ เช่น ระบบฐานข้อมูล (database) แต่ไดนามิกโมเดลมีความสำคัญสำหรับงานที่มีการกระทำโต้ตอบกัน สำหรับปัญหาส่วนมากนั้นความถูกต้องแห่งตรรกวิทยา (logical) ขึ้นกับลำดับของการกระทำโต้ตอบกัน ไม่ใช่เวลาที่แน่นอนของการกระทำโต้ตอบกัน แต่ว่าอย่างไรก็ตามในระบบเวลาจริง (real-time) ต้องการเวลาที่ระบุเฉพาะของการกระทำโต้ตอบกัน ซึ่งจะต้องคิดถึงในขณะวิเคราะห์ แต่ในที่นี้จะไม่พูดถึงการวิเคราะห์เวลาจริง

อันดับแรกจัดเตรียมภาพสถานการณ์ (scenarios) ถึงแม้ว่าอาจจะไม่ครอบคลุมทั้งหมดที่อาจจะเกิดขึ้นได้แต่อย่างน้อยก็มั่นใจได้ว่าการกระทำโต้ตอบแบบธรรมดาจะไม่ถูกมองข้ามไปเหตุการณ์ได้มาจากภาพสถานการณ์ โดยปกติจะเป็นการดีที่สุดที่จะกำหนดอีเวนท์เป็นอันดับแรกและจากนั้นจึงกำหนดแต่ละอีเวนท์ให้กับออบเจคเป้าหมายของอีเวนท์นั้น แล้วจัดลำดับของอีเวนท์และสเตท (state) ลงในสเตทไดอะแกรม (state diagram) สุดท้ายก็เปรียบเทียบสเตทไดอะแกรมสำหรับออบเจคต่างๆ เพื่อให้เกิดความมั่นใจว่าออบเจคแลกเปลี่ยนอีเวนท์เป็นคู่ที่เข้ากันได้ผลลัพธ์ของสเตทไดอะแกรมสร้างขึ้นเป็นไดนามิกโมเดล

ขั้นตอนการสร้างไดนามิกโมเดลมีดังต่อไปนี้คือ

- จัดเตรียมภาพสถานการณ์ของลำดับการกระทำโต้ตอบกัน
- กำหนดอีเวนท์ระหว่างออบเจค
- เตรียมทางเดินของอีเวนท์สำหรับแต่ละภาพสถานการณ์
- สร้างสเตทไดอะแกรม
- จับคู่ระหว่างอีเวนท์กับออบเจคเพื่อตรวจสอบความคงเส้นคงวา

จัดเตรียมบทสนทนาระหว่างผู้ใช้และระบบ เพื่อใช้กำหนดพฤติกรรมของระบบ ภาพสถานการณ์เหล่านี้แสดงการกระทำโต้ตอบหลัก รูปแบบการแสดงต่อภายนอกและการแลกเปลี่ยนข่าวสาร ภาพสถานการณ์ทำให้เข้าใกล้ไดนามิกโมเดลได้มากกว่าการพยายามเขียนแบบ

จำลองแบบทั่วไปโดยตรงและยังรับประกันได้ว่าจะไม่ข้ามขั้นตอนที่สำคัญ รวมทั้งเส้นทางของการกระทำโต้ตอบกันทั้งหมดถูกต้อง

ในบางครั้งประโยคปัญหาอธิบายลำดับการกระทำโต้ตอบกันได้ทั้งหมด แต่เวลาที่ใช้ส่วนมากใช้ในการกำหนดรูปแบบของการกระทำโต้ตอบกัน ประโยคปัญหาอาจจะกำหนดข่าวสารที่ต้องการแต่ไม่ระบุวิธีการได้รับข่าวสาร ในระบบงานหลายระบบการรวบรวมข้อมูลอินพุทคืองานส่วนใหญ่ ไดนามิกโมเดลมีความสำคัญสำหรับระบบงานเช่นนี้

เริ่มต้นเตรียมภาพสถานการณ์สำหรับกรณี "ปกติ" การกระทำโต้ตอบกันที่ไม่มีข้อมูลอินพุทที่ไม่ปกติหรือเงื่อนไขข้อผิดพลาด จากนั้นจึงคิดถึงกรณี "พิเศษ" เช่น ลำดับของข้อมูลอินพุทที่ละเว้นได้ ค่าสูงสุดและต่ำสุด และค่าซ้ำ ต่อไปคิดถึงกรณีข้อผิดพลาดของผู้ใช้ ค่าที่หาค่าไม่ได้และความไม่สำเร็จที่จะต้องตอบ สำหรับระบบงานชนิดมีการกระทำโต้ตอบกันหลายระบบการเก็บข้อผิดพลาดคือส่วนที่รำบากที่สุดของการพัฒนา ถ้าเป็นไปได้ควรยินยอมให้ผู้ใช้ยกเลิกการกระทำนั้นหรือถอยหลังกลับไปตั้งต้นที่จุดที่ไม่มีปัญหาก่อนหน้านั้น ท้ายสุดคิดถึงชนิดของการกระทำโต้ตอบกันหลายชนิดที่สามารถกำหนดเป็นการกระทำโต้ตอบขั้นพื้นฐานได้เช่น การร้องขอความช่วยเหลือ และสถานะของคำถาม(status queries)

ภาพสถานการณ์คือลำดับของอีเวนท์ อีเวนท์เกิดขึ้นเมื่อมีการแลกเปลี่ยนข่าวสารระหว่างออบเจคในระบบและตัวกระทำภายนอกเช่น user, sensor หรืองานอื่น ค่าของข่าวสารที่แลกเปลี่ยนคือพารามิเตอร์ของอีเวนท์ อีเวนท์ที่ไม่มีพารามิเตอร์เป็นอีเวนท์ที่มีความหมายและเป็นแบบธรรมดา ข่าวสารในอีเวนท์คือความจริงที่เกิดขึ้นเป็นสัญญาณ (signal) เมื่อข่าวสารถอินพุทในระบบ หรือเอาว์พุทจากระบบมีอีเวนท์เกิดขึ้น

สำหรับแต่ละอีเวนท์กำหนดผู้กระทำ (เช่น ระบบ ผู้ใช้ หรือผู้กระทำภายนอกอื่นๆ) ที่ทำให้เกิดอีเวนท์และพารามิเตอร์ของอีเวนท์ รูปแบบของจอภาพ หรือรูปแบบการแสดงผลลัพธ์ไม่ได้มีอิทธิพลเหนือการกระทำโต้ตอบกันหรือการแลกเปลี่ยนค่า ไม่ต้องกังวลถึงรูปแบบผลลัพธ์สำหรับการเริ่มต้นออกแบบไดนามิกโมเดล เอาไว้อธิบายตอนขัดเกลาแบบจำลอง

การกระทำโต้ตอบกันสามารถแบ่งออกได้เป็นสองส่วน คือลอจิกของระบบงานและการติดต่อกับผู้ใช้ นักวิเคราะห์ระบบควรเพ่งเล็งเป็นอันดับแรกที่การโฟลของข่าวสารและการควบคุมข่าวสารมากกว่ารูปแบบการแทนค่า ลอจิกของโปรแกรมเดียวกันสามารถจะรับการอินพุทจากบรรทัดของคำสั่ง แฟ้มข้อมูล เมาส์ การสัมผัสจอภาพ กดปุ่มคีย์บอร์ด หรือการเชื่อมต่อระยะไกล ถ้าหากว่ารายละเอียดของการติดต่อแบ่งแยกกันอย่างระมัดระวัง ไดนามิกโมเดลจะเก็บลอจิกการควบคุมของระบบงาน

เป็นการยากที่จะหาวิธีของการติดต่อกับผู้ใช้โดยไม่มีการทดสอบ ลอจิกระบบงานสามารถจำลองขึ้นโดยสร้างเป็นตุ๊กตาของวิธีดำเนินงาน (dummy procedure) ลอจิกของระบบงานที่แยกกันกับการติดต่อกับผู้ใช้ยอมให้หาวิธีการติดต่อกับผู้ใช้ได้ในขณะที่ระบบงานกำลังพัฒนาอยู่

4.4.1 การกำหนดอีเวนท์

ตรวจสอบภาพสถานการณ์เพื่อกำหนดอีเวนท์ภายนอกทั้งหมด ได้แก่ สัญญาณทั้งหมด ข้อมูลอินพุท การตัดสินใจ การขัดจังหวะ การเปลี่ยนสถานะ และการกระทำกับผู้ใช้หรือจากผู้ใช้ การกระทำกับอุปกรณ์ภายนอกหรือจากอุปกรณ์ภายนอก แต่ขั้นตอนการคำนวณภายในไม่ใช่อีเวนท์ยกเว้นสำหรับจุดตัดสินใจซึ่งกระทำโต้ตอบกับโลกภายนอก ใช้ภาพสถานการณ์ค้นหาอีเวนท์ปกติ แต่อย่าลืมกรณีที่เกิดผิดพลาดและอีเวนท์ที่ไม่ปกติ

การกระทำของออบเจคที่ส่งผ่านข่าวสารคืออีเวนท์ ตัวอย่างเช่นในระบบ ATM การใส่รหัสผ่านคืออีเวนท์ส่งจากผู้กระทำภายนอกคือ User ถึง ATM การกระทำโต้ตอบกันระหว่างออบเจคกับออบเจคและโอเปอเรชันตรงกับอีเวนท์ ตัวอย่างเช่น การใส่บัตรคืออีเวนท์ซึ่งส่งจาก User ถึง ATM การโฟลของข่าวสารบางอย่างนั้นมีอยู่ชัดเจน ตัวอย่างเช่น การจ่ายเงินสดคืออีเวนท์จาก ATM ถึง User

อีเวนท์ซึ่งมีผลแบบเดียวกันบนทิศทางของการควบคุมกำหนดให้มีชื่อเดียวกันถึงแม้ว่าค่าพารามิเตอร์จะแตกต่างกัน ตัวอย่างเช่น การใส่รหัสผ่านควรจะเป็นคลาสของอีเวนท์เมื่อค่าของรหัสผ่านไม่มีอิทธิพลบนทิศทางของการควบคุม ในทำนองเดียวกันการจ่ายเงินสดคือคลาสของอีเวนท์ด้วยเมื่อจำนวนเงินไม่ได้มีอิทธิพลบนทิศทางของการควบคุม อีเวนท์ที่มีอิทธิพลเหนือทิศทางของการควบคุมควรจะกำหนดแตกต่างกัน เช่น บัญชีถูกต้อง, บัญชีไม่ถูกต้อง, และรหัสผ่านผิดคืออีเวนท์ที่แตกต่างกัน อย่ารวมกลุ่มเข้าด้วยกันโดยใช้สถานะของบัตร

ถ้าค่าที่เป็นปริมาณแตกต่างกันมีความสำคัญเพียงพอให้กำหนดเป็นอีเวนท์ที่มีความแตกต่าง ตัวอย่างเช่น การกดปุ่มตัวเลขแต่ละตัวที่แตกต่างกันปกติจะคิดว่าเป็นอีเวนท์เดียวกันเมื่อการควบคุมไม่ได้ขึ้นกับค่าของตัวเลข การกดปุ่ม "enter" อาจจะคิดว่าเป็นอีเวนท์ที่แตกต่างกันเมื่อระบบงานตอบสนองแตกต่างกัน การกำหนดความแตกต่างกันขึ้นกับประเภทระบบงาน เราต้องสร้างสเตทไดอะแกรมก่อนจึงจะสามารถกำหนดอีเวนท์ทั้งหมดได้ ความแตกต่างระหว่างอีเวนท์บางอย่างจะไม่มีผลกับพฤติกรรมดังนั้นไม่เอาใจใส่ได้

กำหนดอีเวนท์แต่ละชนิดให้กับคลาสซึ่งส่งและรับอีเวนท์นั้น อีเวนท์เป็นแบบอินพุทสำหรับผู้รับและเป็นแบบเอาว์พุทสำหรับผู้ส่ง บางครั้งออบเจคส่งอีเวนท์ให้กับตนเอง ในกรณีนี้อีเวนท์จะเป็นทั้งอินพุทและเอาว์พุทสำหรับคลาสเดียวกัน

แสดงแต่ละภาพสถานการณ์เหมือนเป็นทางเดินของอีเวนท์ รายการคำสั่งของอีเวนท์ระหว่างออบเจคต่างๆกำหนดในคอลัมน์ของตาราง ถ้ามีออบเจคมากกว่าหนึ่งออบเจคของคลาสที่มีส่วนร่วมในภาพสถานการณ์เดียวกัน กำหนดเป็นคอลัมน์ที่แยกกันสำหรับแต่ละออบเจคโดยการพิจารณาคอลัมน์ที่ระบุเฉพาะในทางเดิน เราสามารถมองเห็นอีเวนท์ซึ่งมีอิทธิพลโดยตรงเหนือออบเจคที่ระบุเฉพาะได้เฉพาะอีเวนท์เหล่านี้เท่านั้น ซึ่งจะปรากฏในสเตทไดอะแกรมสำหรับออบเจค

แผนผังการโฟลของอีเวนท์ (event flow diagram) แสดงอีเวนท์ระหว่างกลุ่มของคลาส (เช่น โมดูล) แผนผังนี้สรุปอีเวนท์ระหว่างคลาสโดยไม่สนใจการเรียงลำดับคือรวมอีเวนท์จากภาพสถานการณ์ทั้งหมดและอีเวนท์ของความผิดพลาด แผนผังการโฟลของอีเวนท์คือสำเนาที่เปลี่ยนแปลงอยู่เสมอของออบเจคไดอะแกรม เส้นทางในออบเจคไดอะแกรมแสดงการโฟลของข่าวสารที่เป็นไปได้ แต่เส้นทางในแผนผังการโฟลของอีเวนท์แสดงการโฟลของการควบคุมที่เป็นไปได้

4.4.2 การสร้างสเตทไดอะแกรม

เตรียมสเตทไดอะแกรมสำหรับแต่ละคลาสกับพฤติกรรมที่เปลี่ยนแปลงอยู่เสมอแสดงอีเวนท์ที่ออบเจครับและส่ง ทุกภาพสถานการณ์หรือทางเดินของอีเวนท์ตรงกับเส้นทางในสเตทไดอะแกรม แต่ละกิ่งในการโฟลของการควบคุมถูกแทนที่โดยสเตทซึ่งการเปลี่ยนสเตทมีทางออกมากกว่าหนึ่งทาง เริ่มต้นด้วยแผนผังเส้นทางอีเวนท์ซึ่งมีอิทธิพลเหนือคลาสที่จะจำลองขึ้น เลือกเส้นทางที่แสดงการกระทำโต้ตอบกันและอีเวนท์ที่มีอิทธิพลเหนือออบเจคเพียงออบเจคเดียว จัดอีเวนท์เข้าไปในเส้นทางและเขียนชื่อของอีเวนท์ที่เป็นแบบอินพุทหรือเอาว์พุทซึ่งได้มาจากคอลัมน์ในแผนผังเส้นทางอีเวนท์ สเตทจะอยู่ระหว่างสองอีเวนท์ ใส่ชื่อของสเตทแต่ละสเตท ถ้าชื่อมีความหมายแต่ถ้าไม่มีก็ไม่ต้องกังวล แผนผังเริ่มแรกจะเป็นลำดับของอีเวนท์และสเตท ถ้าภาพสถานการณ์สามารถทำซ้ำได้อย่างไม่จำกัดให้ปิดเส้นทางในสเตทไดอะแกรม

หาวงรอบในแผนผัง ถ้าลำดับของอีเวนท์สามารถทำซ้ำได้อย่างไม่จำกัดดังนั้นมันจะอยู่ในรูปของวงรอบ (loop) แทนที่ลำดับของอีเวนท์ด้วยวงรอบเมื่อเป็นไปได้ ในวงรอบสเตทแรกและสเตทสุดท้ายเหมือนกันทุกอย่าง แต่ถ้าออบเจค "จำได้" ว่ามันได้ข้ามวงรอบ ดังนั้นสองสเตท

ไม่ได้เหมือนกันทุกอย่างจริงและวงรอบแบบธรรมดานั้นไม่ถูกต้อง อย่างน้อยที่สุดสเตทในวงรอบต้องมีหลายทรานแซคชัน (transaction) ออกจากออบเจคหรือไม่เช่นนั้นวงรอบนั้นจะไม่มีวันเลิกวนรอบ

รวมภาพสถานการณ์อื่นๆ เข้าในสเตทไดอะแกรม หาจุดในแต่ละภาพสถานการณ์ซึ่งมันแผ่ออกจากภาพสถานการณ์ก่อนหน้านี้ จุดนี้ตรงกับสเตทที่มีอยู่แล้วในไดอะแกรม อีเวนท์ใหม่ที่เกิดตามหลังสเตทที่มีอยู่แล้วคือเส้นทางที่ให้เลือกได้ ในขณะที่ตรวจสอบสเตทและภาพสถานการณ์ เราอาจจะคิดถึงอีเวนท์ที่เป็นไปได้อื่นๆ ซึ่งสามารถเกิดขึ้นได้ที่แต่ละสเตท เพิ่มอีเวนท์เหล่านั้นเข้าไปในสเตทไดอะแกรมด้วย

สิ่งที่ยากที่สุดคือการตัดสินใจว่าสเตทไหนที่เส้นทางที่เลือกได้มารวมกับไดอะแกรมที่มีอยู่แล้ว เส้นทางสองเส้นทางมารวมกันที่สเตทถ้าออบเจค "ลืม" ว่าเป็นเส้นทางไหนในหลายกรณีจากระบบงานปรากฏอย่างชัดเจนว่าสองสเตทเหมือนกันทุกอย่าง ตัวอย่างเช่น ใส่ 2 nikels ในเครื่องขายของเท่ากับใส่ 1 dime

ระวังเส้นทางสองเส้นทางที่ปรากฏเหมือนกันทุกอย่าง แต่สามารถที่จะแยกความแตกต่างได้ภายใต้พฤติกรรมบางอย่าง ตัวอย่างเช่น บางระบบทำซ้ำลำดับการใส่ข่าวสารถ้าผู้ใช้ใส่ผิดจะยกเลิกการทำงานหลังจำนวนครั้งจำนวนหนึ่งที่ทำผิด ลำดับการทำซ้ำเกือบเหมือนกันเว้นแต่มันจำความผิดพลาดที่ผ่านไปได้ ความแตกต่างสามารถทำได้โดยการเพิ่มพารามิเตอร์ไว้จำข่าวสาร เช่นจำนวนความผิดพลาด อย่างน้อยที่สุดการเปลี่ยนสเตทต้องขึ้นกับค่าของพารามิเตอร์ การใช้พารามิเตอร์ที่ฉลาดและการเปลี่ยนสเตทโดยมีเงื่อนไขสามารถทำให้สเตทไดอะแกรมง่ายลงอย่างมากมาย แต่ต้องเสียค่าใช้จ่ายในการรวมข่าวสารของสเตทและข้อมูล สเตทไดอะแกรมที่ข้อมูลมีความไม่เป็นอิสระมากจนเกินไปจะทำให้สับสนและไม่เป็นดั่งที่คิด สิ่งที่ให้เลือกอื่นๆ คือการแบ่งสเตทไดอะแกรมเป็นสองซับไดอะแกรม (subdiagram) ที่เกิดขึ้นพร้อมๆกัน ใช้ส่วนหนึ่งสำหรับแนวทางหลักและอีกส่วนสำหรับแยกความแตกต่างข่าวสาร ตัวอย่างเช่น ซับไดอะแกรมที่ยอมสำหรับความผิดพลาดของผู้ใช้อาจจะมีสเตทเป็นไม่มีข้อผิดพลาดและมีข้อผิดพลาด

ภายหลังที่คิดถึงอีเวนท์ปกติแล้วเพิ่มกรณีที่อยู่ในขอบเขตและกรณีพิเศษ คิดถึงอีเวนท์ที่เกิดขึ้นล่าช้า ตัวอย่างเช่น การร้องขอเพื่อจะยกเลิกแทรนแซคชันภายหลังที่ได้ส่งเข้ามาสำหรับการทำงานแล้วในกรณีเมื่อผู้ใช้ (หรือผู้กระทำภายนอกอื่นๆ) อาจจะไม่ได้รับการตอบสนองและทรัพยากร (resource) บางอย่างต้องถูกเรียกคืน และอีเวนท์นั้นหมดเวลา การจับข้อผิดพลาดของผู้ใช้มักจะต้องการความคิดและการบันทึกมากกว่ากรณีปกติ การจับข้อผิดพลาดมักจะทำให้โครงสร้างของโปรแกรมสับสนแต่จำเป็นต้องทำ

สเตทไดอะแกรมของคลาสทำสำเร็จเมื่อไดอะแกรมครอบคลุมภาพสถานการณ์ทั้งหมดและไดอะแกรมเก็บอีเวนท์ทั้งหมดที่สามารถมีอิทธิพลเหนือออบเจคของคลาสในแต่ละสเตทของออบเจค เราสามารถใช้สเตทไดอะแกรมเสนอแนะภาพสถานการณ์ใหม่โดยการคิดว่าบางอีเวนท์ที่ไม่ได้มีอยู่ในไดอะแกรมจะมีอิทธิพลเหนือสเตทของออบเจคอย่างไร การวางคำถามว่า 'อะไรถ้า' คือวิธีที่ดีที่จะทดสอบความครบถ้วนและขีดความสามารถในการจับข้อผิดพลาดของคลาส (และอาจจะทำซ้ำที่ในระดับโมดูลและระบบได้ด้วย)

ถ้ามีการกระทำโต้ตอบกันที่ซับซ้อนกับการอินพุทที่ไม่ขึ้นต่อกันเราสามารถจัดไดนามิกโมเดลโดยใช้ nested state diagram หรือมิฉะนั้นสเตทไดอะแกรมแบบธรรมดาก็เพียงพอแล้วถ้ามันไม่ซับซ้อน

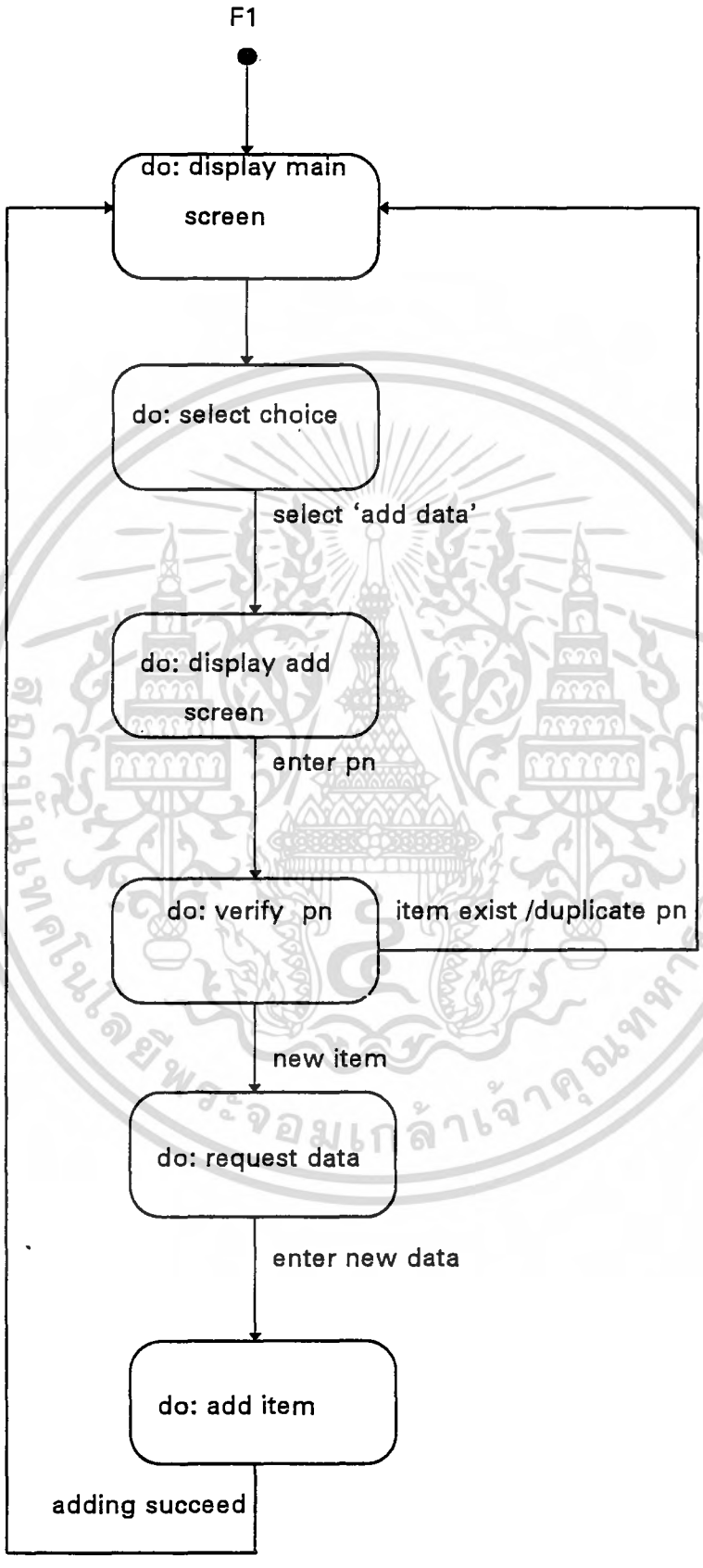
ให้ทำซ้ำตามวิธีการสร้างสเตทไดอะแกรมที่กล่าวมาข้างต้น สำหรับแต่ละคลาสของออบเจค เพ่งเล็งเฉพาะคลาสที่มีการกระทำโต้ตอบที่สำคัญ ไม่ต้องเขียนสเตทไดอะแกรมสำหรับทุกคลาส ออบเจคหลายออบเจคที่ตอบสนองต่ออีเวนท์ที่เป็นอินพุทอย่างเป็นอิสระ ประวัติที่ผ่านมาของออบเจคที่ไม่ได้มีอิทธิพลเหนือการควบคุมคือพารามิเตอร์ ออบเจคอาจจะรับหรือส่งอีเวนท์ เขียนรายชื่อของอีเวนท์ที่เป็นแบบอินพุทสำหรับแต่ละออบเจค และอีเวนท์ที่เป็นแบบเอาท์พุทส่งไปตอบสนองต่ออีเวนท์ที่เป็นแบบอินพุท

ในที่สุดเราอาจจะเขียนสเตทไดอะแกรมได้โดยไม่ต้องเตรียมเส้นทางอีเวนท์ โดยปกติภาพสถานการณ์เพียงเล็กน้อยก็มีประโยชน์ในบางกรณี

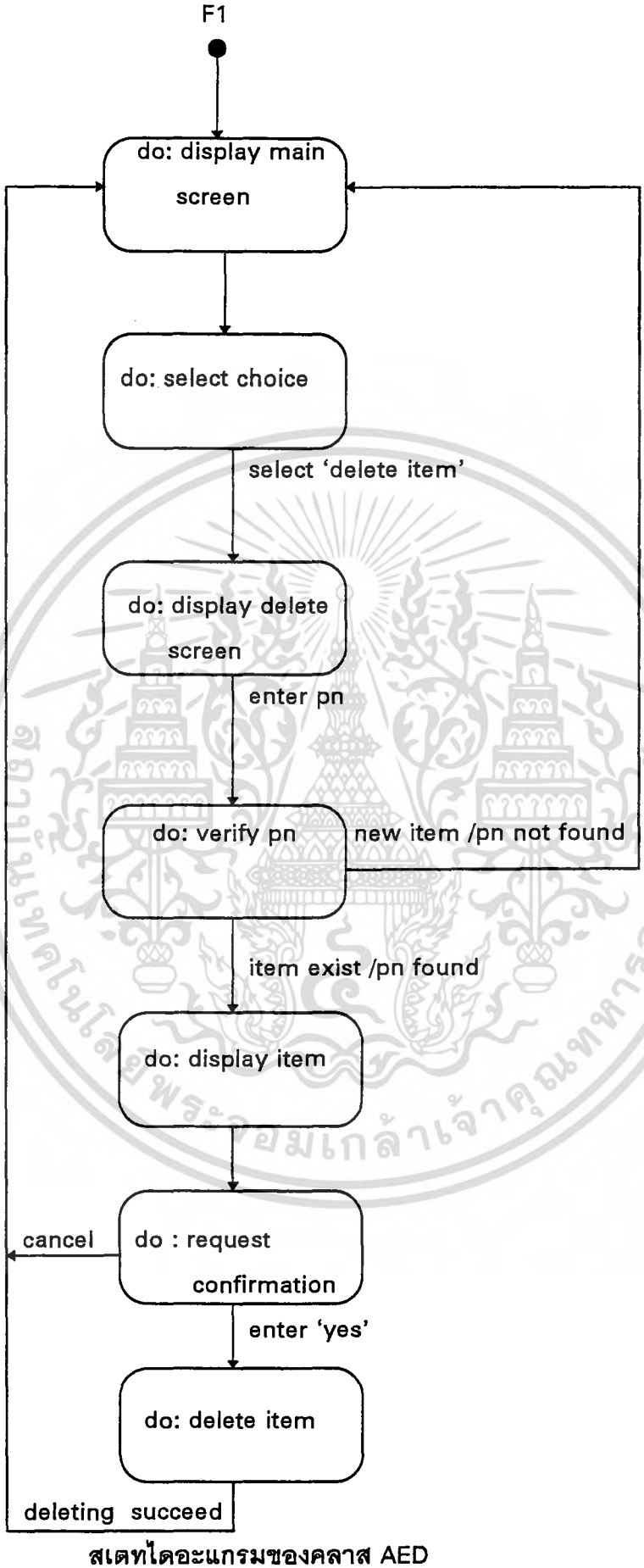
ตรวจสอบความครบถ้วนและคงเส้นคงวาเมื่อสร้างสเตทไดอะแกรมสำหรับแต่ละคลาสเสร็จ ทุกๆอีเวนท์ต้องมีผู้ส่งและผู้รับ บางครั้งทั้งผู้ส่งและผู้รับคือคลาสเดียวกัน สเตทที่ไม่มีผู้ให้กำเนิดและผู้รับต่อนี้น่าสงสัยถ้ามั่นใจว่าสเตทเหล่านั้นไม่ใช่จุดเริ่มต้นหรือจุดจบของลำดับการกระทำโต้ตอบกัน ติดตามผลของอีเวนท์ที่อินพุทจากออบเจคถึงออบเจคตลอดทั้งระบบ เพื่อให้เกิดความมั่นใจว่ามันเข้ากันได้กับภาพสถานการณ์ ออบเจคที่เกิดขึ้นพร้อมๆ กันให้ระวังข้อผิดพลาดของการทำพร้อมกันเมื่อมีอินพุทที่ล่าช้าไป ต้องมั่นใจว่าการตรงกันของอีเวนท์บนสเตทไดอะแกรมต่างๆ มีความคงเส้นคงวา กลุ่มของสเตทไดอะแกรมสำหรับคลาสที่มีพฤติกรรมสำคัญที่เปลี่ยนแปลงอยู่เสมอสร้างไดนามิกโมเดลสำหรับระบบงาน

รายละเอียดของสเตทไดอะแกรมสำหรับออบเจคแต่ละออบเจคในระบบงานบริหารพัสดุสายช่างอากาศแสดงอยู่ในภาพที่ 35

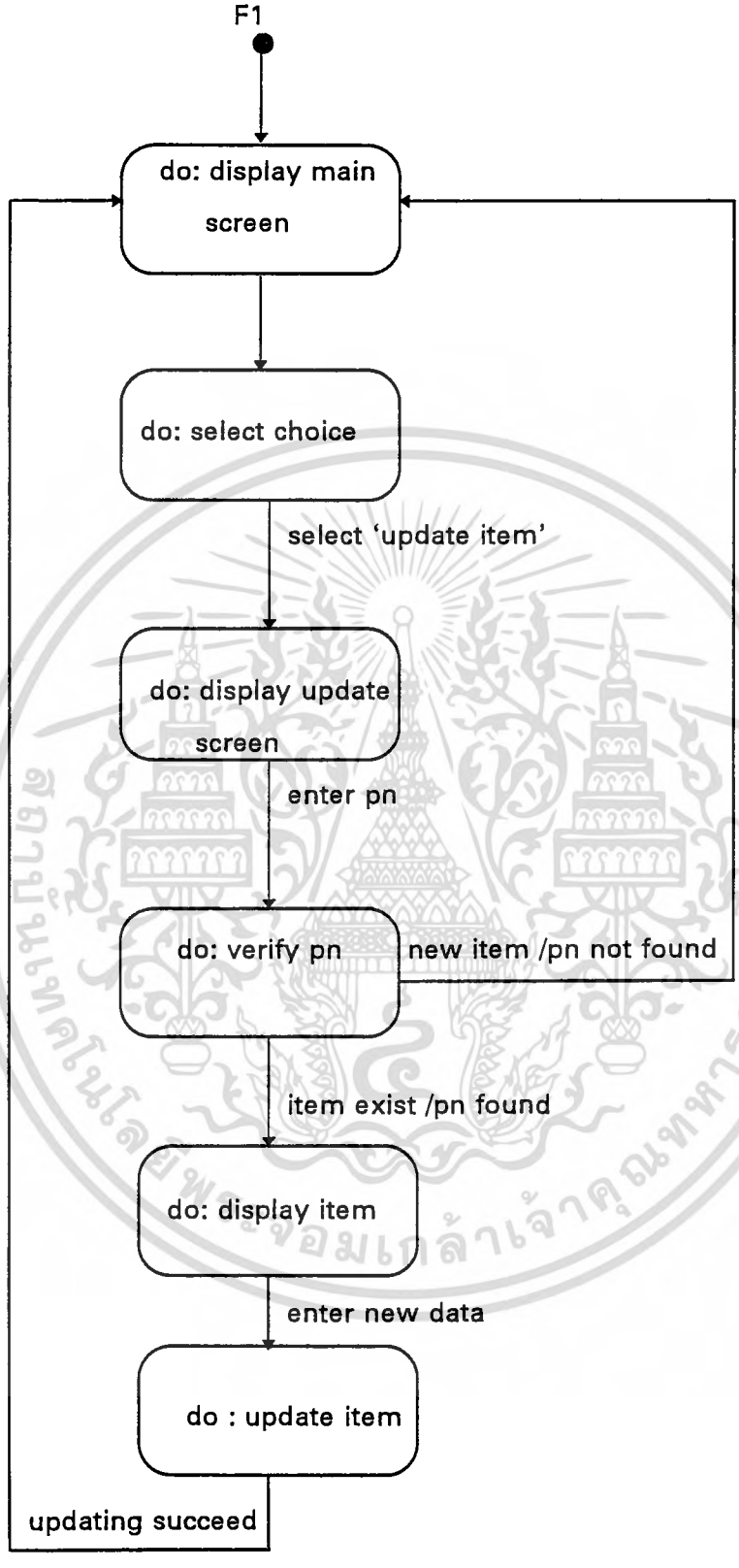
ภาพที่ 35 ไดนามิกโมเดล



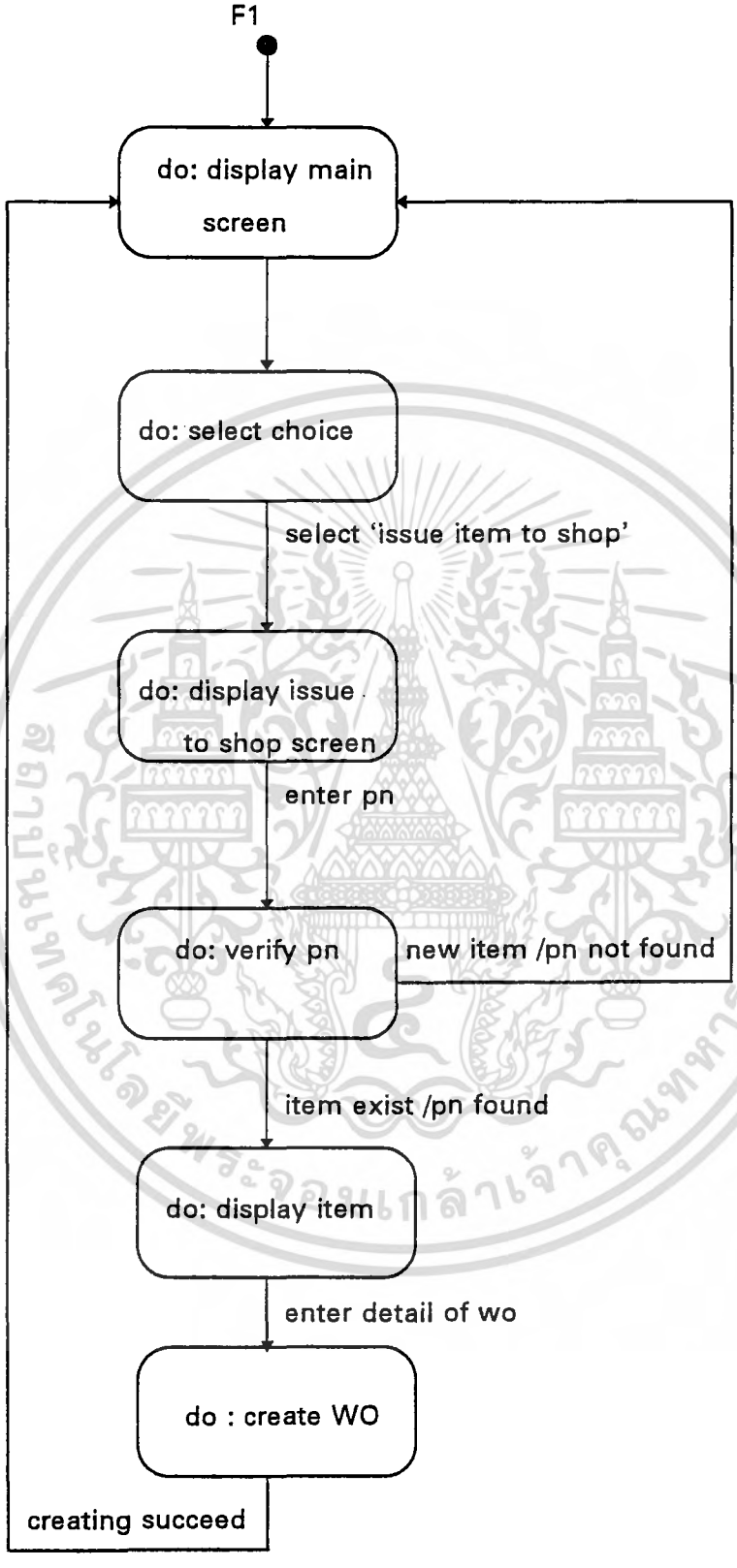
สเตทไดอะแกรมของคลาส AED



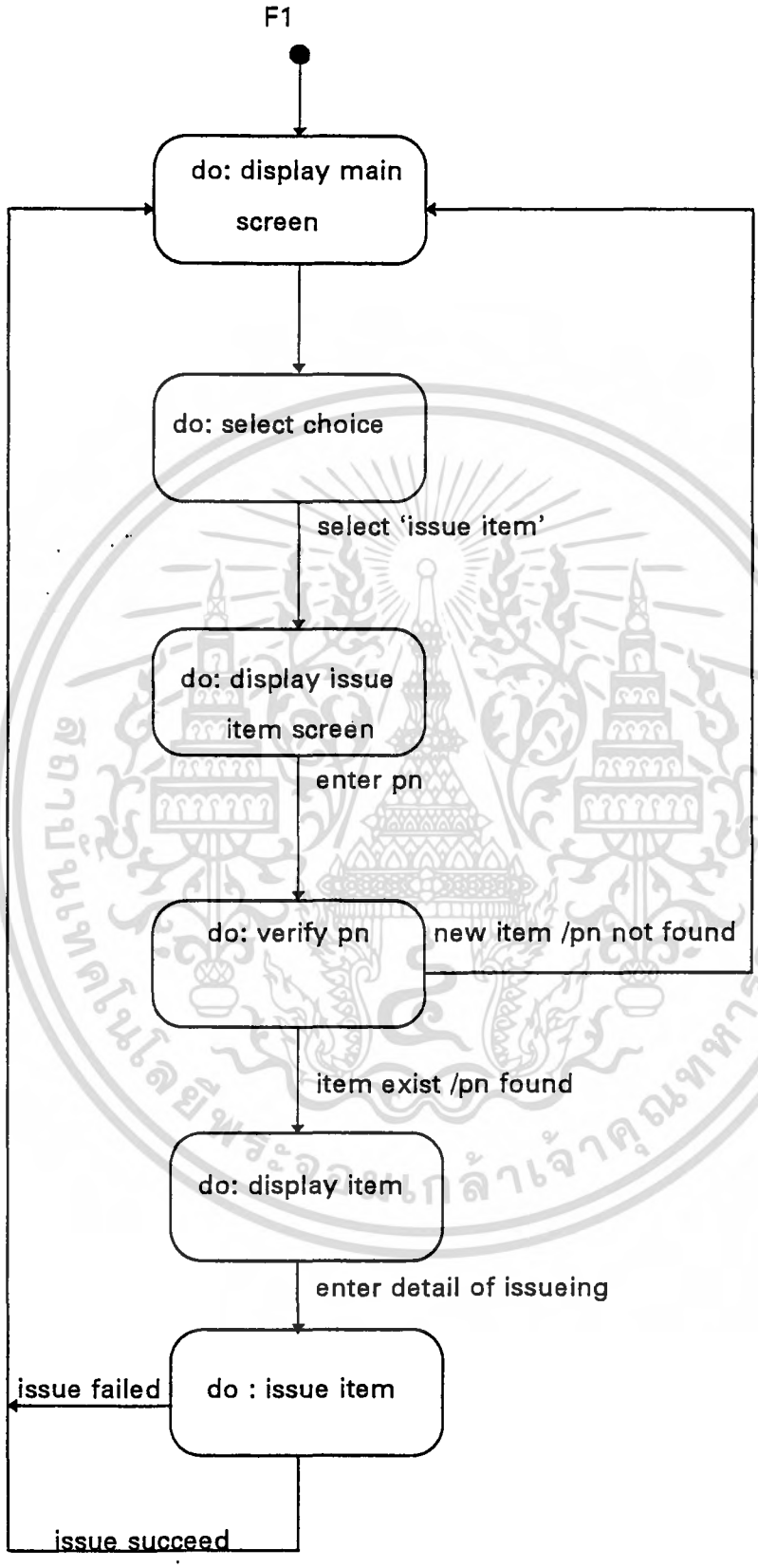
สเตทไดอะแกรมของคลาส AED



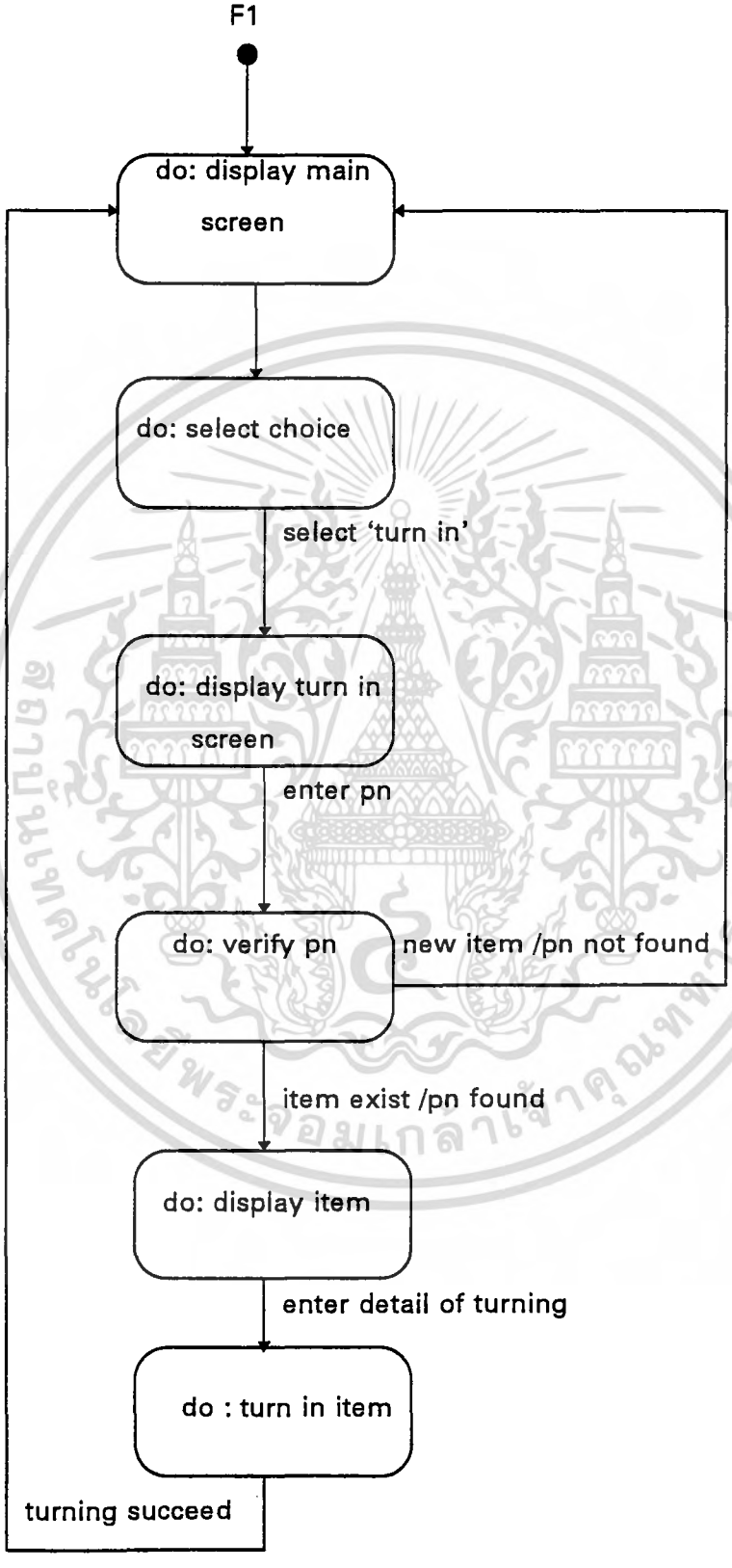
สเตทไดอะแกรมของคลาส AED



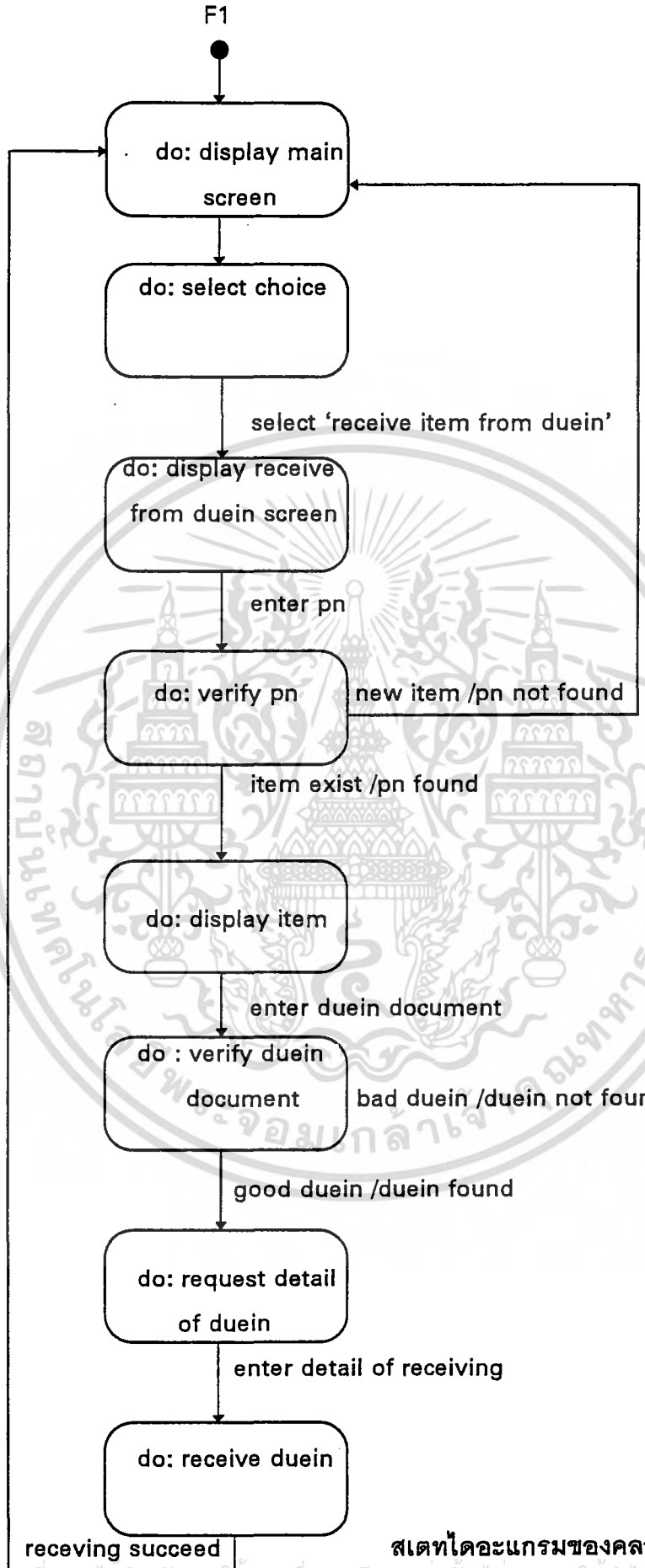
สเตทไดอะแกรมของคลาส AED



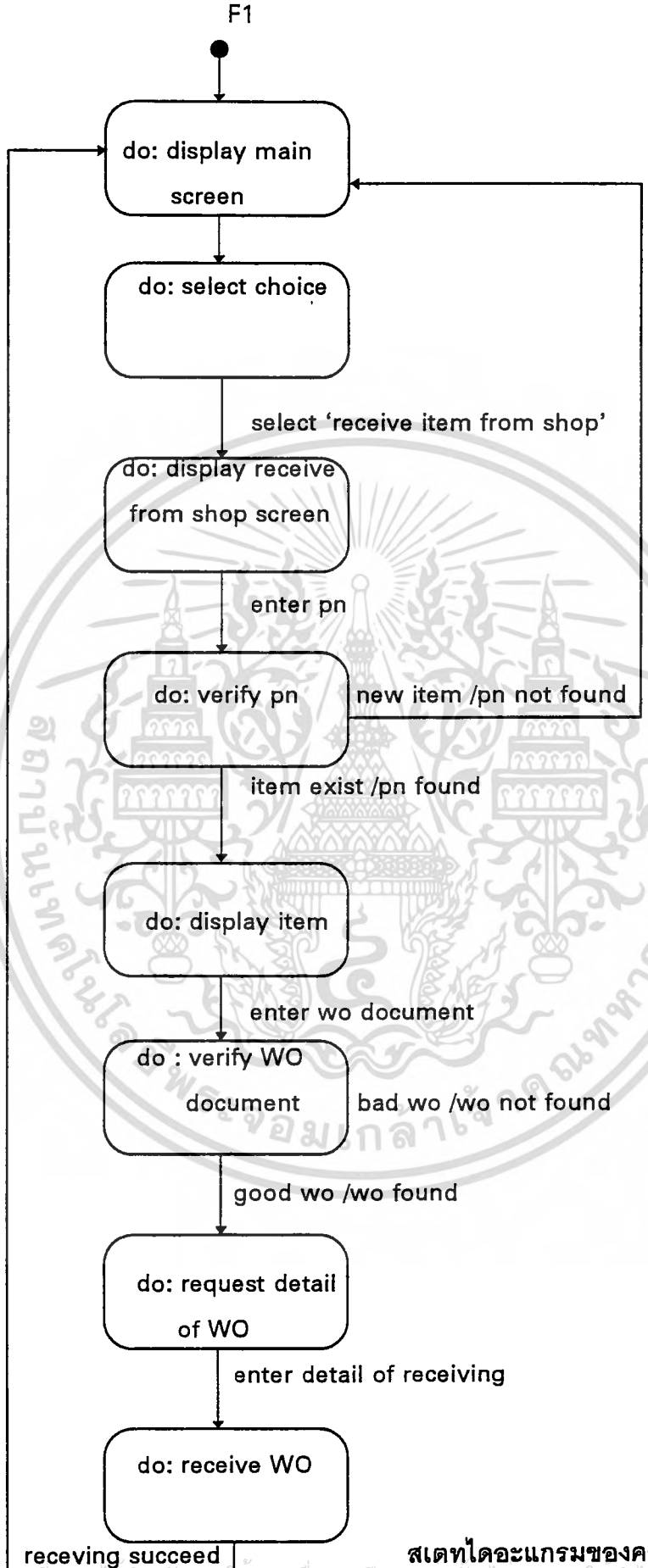
สเตทไดอะแกรมของคลาส AED



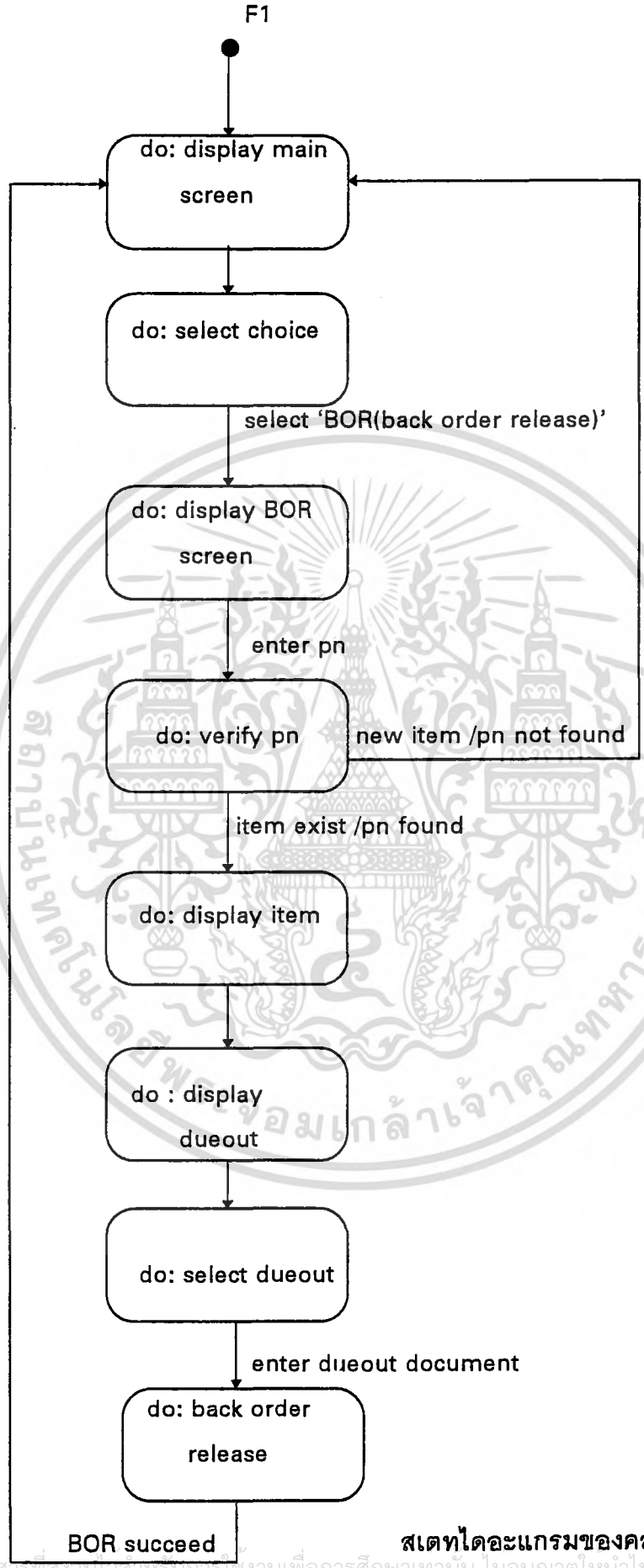
สเตทไดอะแกรมของคลาส AED



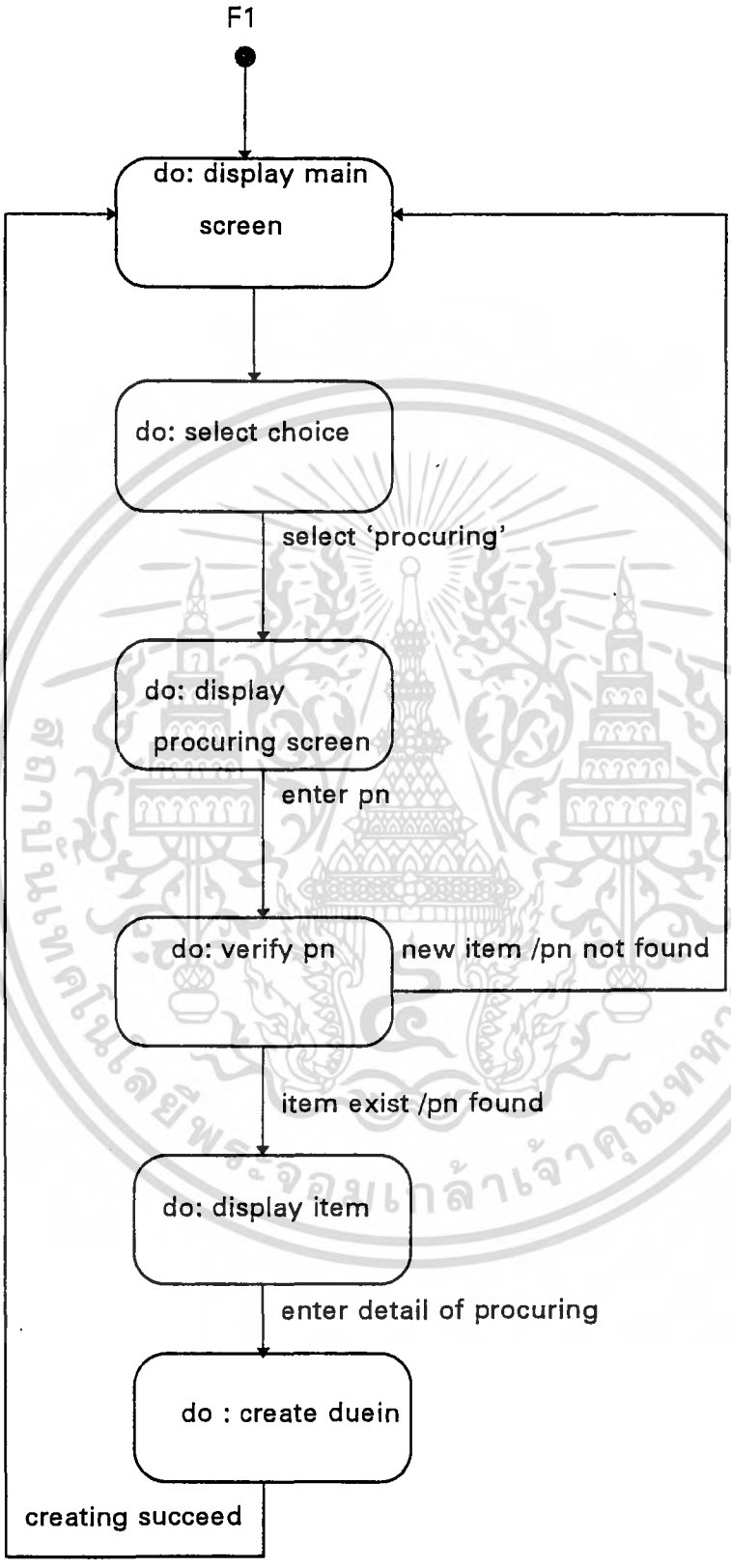
สเตทไดอะแกรมของคลาส AED



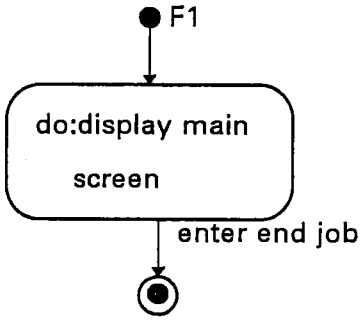
สเตทไดอะแกรมของคลาส AED



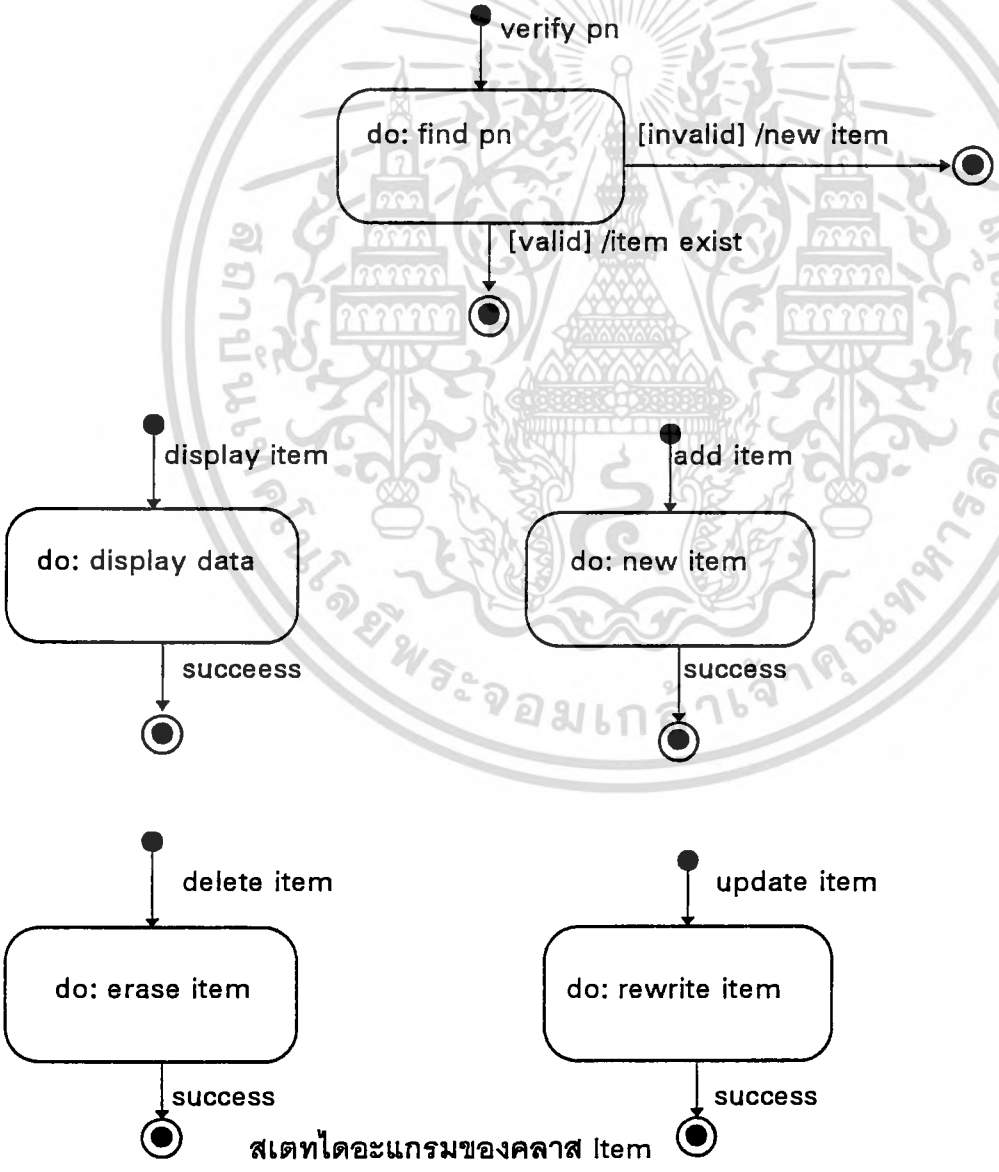
สเตทไดอะแกรมของคลาส AED



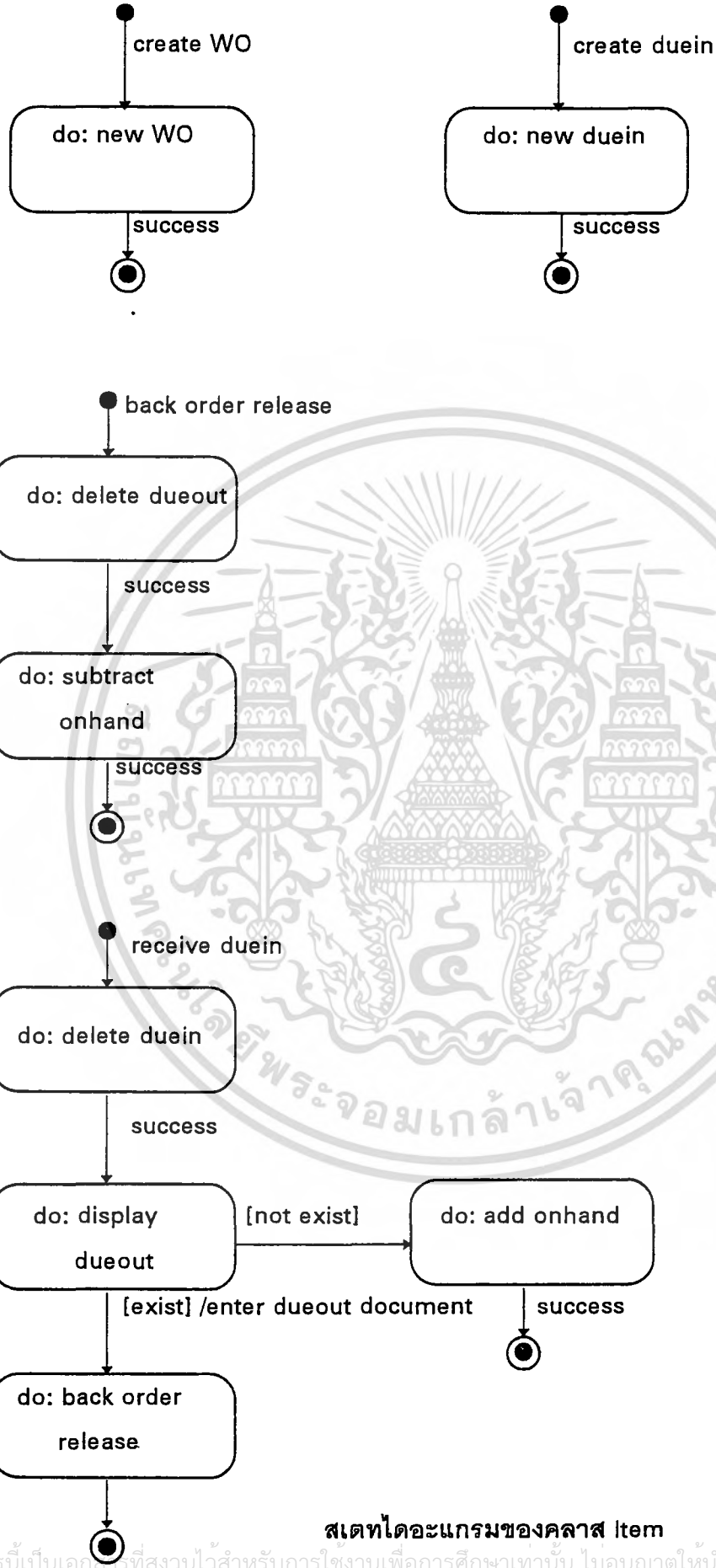
สเตทไดอะแกรมของคลาส AED



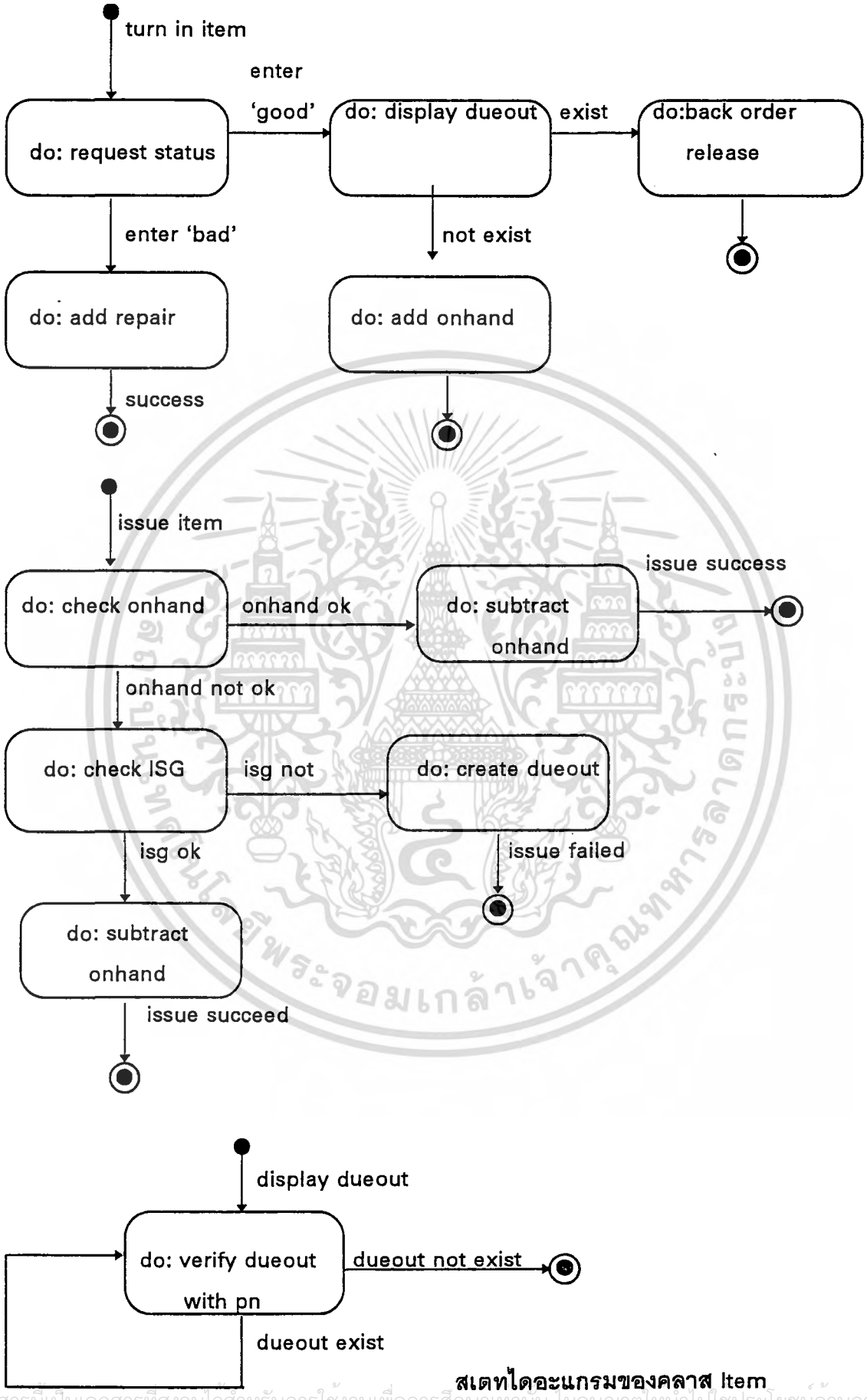
สเตทไดอะแกรมของคลาส AED



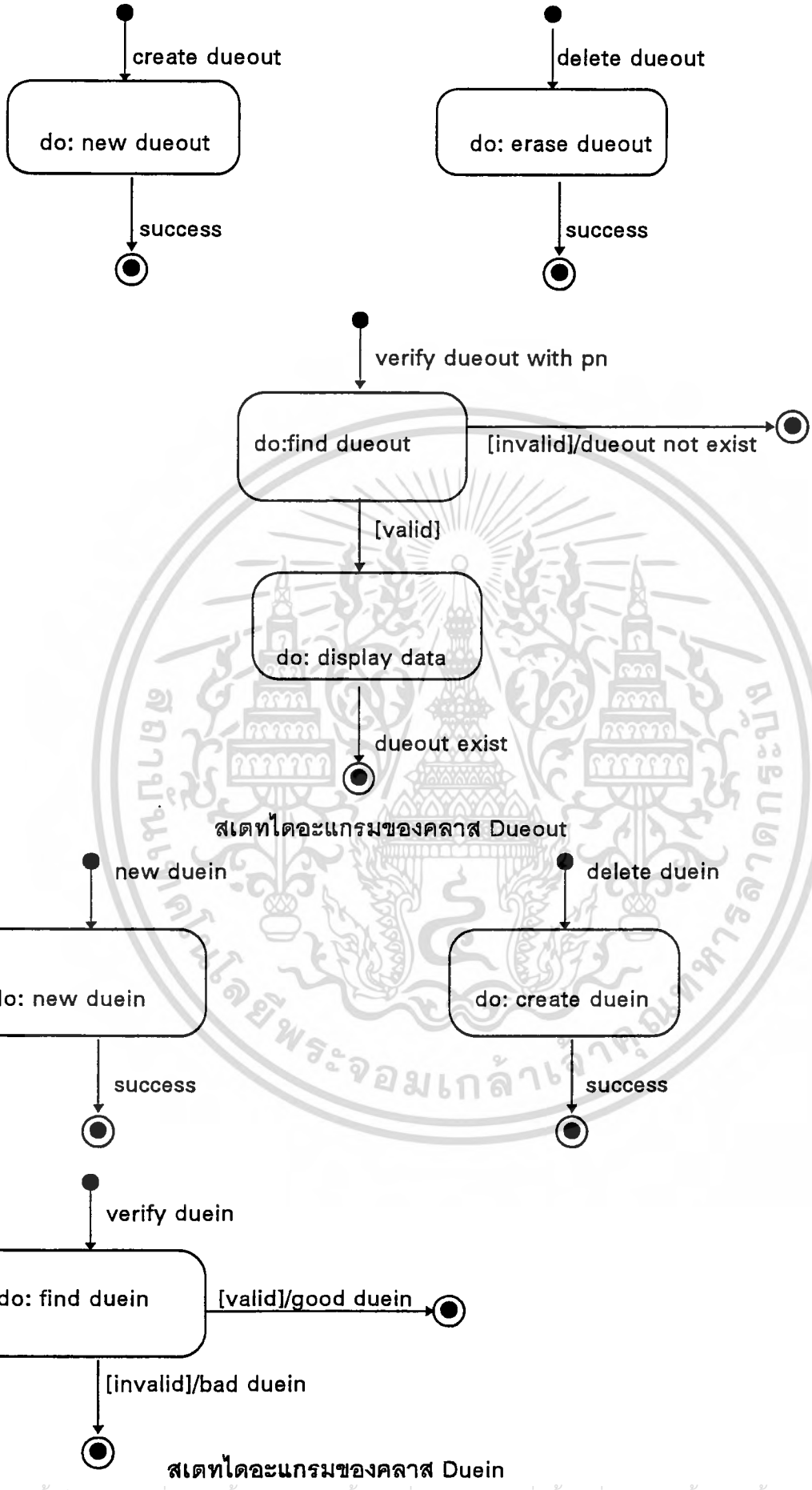
สเตทไดอะแกรมของคลาส Item



สเตทไดอะแกรมของคลาส Item

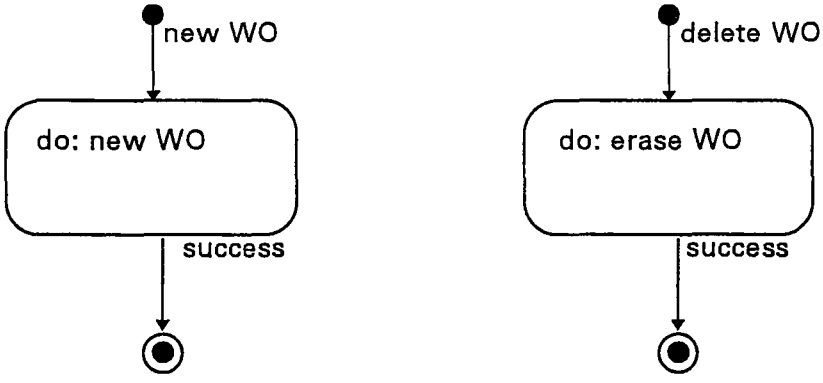


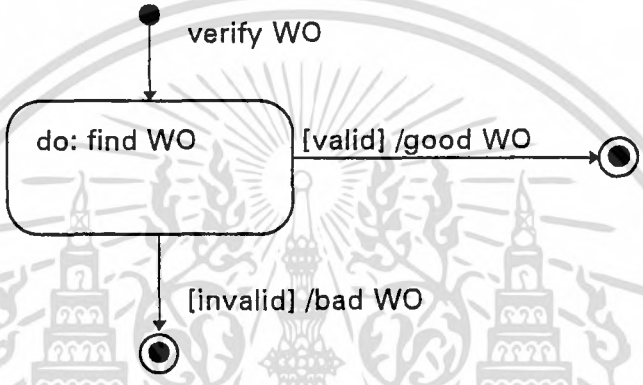
สเตทไดอะแกรมของคลาส Item



สเตทไดอะแกรมของคลาส Dueout

สเตทไดอะแกรมของคลาส Duein





สเตทไดอะแกรมของคลาส WO

4.5 ฟังก์ชันนัลโมเดลลิ่ง (Functional modelling)

ฟังก์ชันนัลโมเดลแสดงการคำนวณค่าว่าได้มาอย่างไร โดยไม่สนใจลำดับ การตัดสินใจ, หรือโครงสร้างของออบเจค ฟังก์ชันนัลโมเดลแสดงค่าซึ่งขึ้นกับค่าและการกระทำอื่นๆ ที่มีความเกี่ยวพันกัน ดาต้าโฟลไดอะแกรม(Data flow diagram) มีประโยชน์สำหรับแสดงการกระทำที่ไม่มีอิสระ การกระทำแสดงได้หลายทาง ได้แก่ ภาษาธรรมชาติ สมการทางคณิตศาสตร์ และซูโดโคด (pseudocode)

โพรเซส(process) ในดาต้าโฟลไดอะแกรมตรงกับแอคทิวิตี้หรือแอคชันในสเตทไดอะแกรมของคลาส การโฟล (flow) บนดาต้าโฟลไดอะแกรมตรงกับออบเจคหรือค่าของแอทริบิวท์ในออบเจคไดอะแกรม วิธีที่ดีที่สุดคือสร้างฟังก์ชันนัลโมเดลภายหลังสร้างออบเจคและไดนามิกโมเดลเสร็จแล้ว

ขั้นตอนการสร้างฟังก์ชันนัลโมเดลมีดังต่อไปนี้

- กำหนดอินพุทและเอาว์พุท
- สร้างดาต้าโฟลไดอะแกรมแสดงความขึ้นต่อกันของฟังก์ชัน
- กำหนดข้อบังคับ (constrains)

4.5.1 การกำหนดอินพุทและเอาว์พุท

เริ่มต้นโดยการเขียนรายการอินพุทและเอาว์พุท อินพุทและเอาว์พุทคือพารามิเตอร์ของอีเวนท์ระหว่างระบบกับโลกภายนอก ตรวจสอบประโยคปัญหาเพื่อหาค่าอินพุทหรือค่าที่เราทำขาดหายไป

4.5.2 สร้างดาต้าโฟลไดอะแกรม

สร้างดาต้าโฟลไดอะแกรม แสดงเอาว์พุทว่าถูกคำนวณจากอินพุทได้อย่างไรดาต้าโฟลไดอะแกรมโดยปกติจะถูกสร้างเป็นชั้น ชั้นบนสุดอาจจะประกอบด้วยโพรเซสเดี่ยวหรือบางทีหนึ่งโพรเซสรวมอินพุท คำนวณค่า และสร้างผลลัพธ์ ในแต่ละชั้นของดาต้าโฟลไดอะแกรมใช้วิธีการทำงานถอยหลังจากแต่ละเอาว์พุทเพื่อแสดงฟังก์ชันที่คำนวณเอาว์พุท ถ้าการอินพุทไปยังโอเปอเรชันคือการอินพุททั้งหมดของทั้งโปรแกรมที่เราทำ มิฉะนั้นบางส่วนของอินพุทที่เป็นโอเปอเรชันคือค่าที่อยู่ระหว่างกลางซึ่งต้องถูกลากเส้นถอยหลัง เราสามารถใช้วิธีการเดินหน้าจากการอินพุทถึงการเอาว์พุทได้ แต่ปกติมันยากที่จะกำหนดการใช้การอินพุททั้งหมดมากกว่าที่จะกำหนดทรัพยากรทั้งหมดของการเอาว์พุท

กระจายแต่โพรเซสที่มีขนาดใหญ่ในโปรแกรมระดับบนสุดไปเป็นดาต้าโฟลไดอะแกรมระดับต่ำกว่า ถ้าโพรเซสระดับที่สองยังคงประกอบด้วยโพรเซสที่มีขนาดใหญ่อยู่สามารถกระจายออกไปได้อีก

ระบบส่วนมากมีออบเจคภายในที่เป็นที่เก็บข้อมูล ซึ่งเก็บค่าไว้ได้ระหว่างการทำซ้ำ ที่เก็บข้อมูลภายในสามารถที่จะแยกความแตกต่างกับการโฟลของข้อมูลหรือโพรเซสได้ เพราะว่าที่เก็บรับข้อมูลซึ่งไม่ได้ให้ผลลัพธ์เป็นการเอาว์พุทในทันทีทันใดแต่ถูกใช้ในเวลาที่ต่างกันในอนาคต

ดาต้าโฟลไดอะแกรมกำหนดความไม่ขึ้นต่อกันระหว่างโอเปอเรเตอร์เท่านั้น ไม่ได้แสดงการตัดสินใจ หรือลำดับของโอเปอเรเตอร์ ในความเป็นจริงบางโอเปอเรเตอร์สามารถจะให้เลือกได้หรือไม่ได้เกิดขึ้นพร้อมๆ กัน ตัวอย่างเช่น รหัสผ่านต้องถูกตรวจสอบก่อนที่จะแก้ไขบัญชี ถ้ารหัสไม่ถูกต้องดังนั้นจะไม่แก้ไขบัญชี ลำดับการตัดสินใจนี้เป็นส่วนหนึ่งของไดนามิกโมเดลไม่ใช่ฟังก์ชันนัลโมเดล

ค่าของข้อมูลบางค่ามีอิทธิพลเหนือการตัดสินใจในไดนามิกโมเดล การตัดสินใจไม่ได้มีอิทธิพลเหนือเอาว์พุทโดยตรงในดาต้าโฟลโมเดล ในขณะที่ดาต้าโฟลโมเดลแสดงเส้นทางเกี่ยวกับการคำนวณที่เป็นไปได้ แต่อย่างไรก็ตามสามารถหาฟังก์ชันเกี่ยวกับการตัดสินใจได้ในดาต้าโฟลโมเดลเมื่อฟังก์ชันเหล่านั้นอาจจะเป็นฟังก์ชันที่สับสนของอินพุท ฟังก์ชันเกี่ยวกับการตัดสินใจสามารถถูกแสดงบนดาต้าโฟลไดอะแกรม แต่เอาว์พุทของฟังก์ชันเกี่ยวกับการตัดสินใจคือสัญญาณควบคุม ระบุด้วยลูกศรเอาว์พุทเป็นเส้นประ ฟังก์ชันเหล่านี้เป็น "ข้อมูลจมลง" ภายในดาต้าโฟลไดอะแกรม เอาว์พุทของฟังก์ชันเหล่านั้นมีอิทธิพลเหนือการโฟลของการควบคุมในไดนามิกโมเดลและไม่ใช่เอาว์พุทโดยตรง ตัวอย่างเช่น การตรวจสอบรหัสผ่านคือฟังก์ชันเกี่ยวกับการตัดสินใจ เราแสดงสัญญาณข้อผิดพลาดซึ่งอาจจะถูกสร้างขึ้นมา แต่ปล่อยให้ลากลูกศรควบคุมถึงโพรเซสแก้ไขบัญชี ถ้าเราต้องการเราก็สามารถที่จะลากเส้นลูกศรควบคุมถึงโพรเซสที่ถูกควบคุมโดยการตัดสินใจได้

เมื่อดาค้าโฟลไดอะแกรมถูกขัดเกลาจนเพียงพอแล้ว เขียนคำอธิบายของแต่ละฟังก์ชัน คำอธิบายสามารถเขียนเป็นภาษาธรรมชาติ สมการคณิตศาสตร์ ซูโดโคด ตารางการตัดสินใจ หรือรูปแบบอื่นๆ ที่เหมาะสม โดยเพ่งเล็งที่ฟังก์ชันทำอะไร ไม่ใช่ว่าจะทำให้เป็นจริงขึ้นได้อย่างไร คำอธิบายจะเป็นแบบบอกเล่าหรือแบบวิธีดำเนินการ คำอธิบายแบบบอกเล่ากำหนดความสัมพันธ์ระหว่างอินพุทและเอาว์พุท และความสัมพันธ์ระหว่างเอาว์พุท ตัวอย่างเช่น คำอธิบายของฟังก์ชัน "เรียงลำดับและลบค่าที่ซ้ำกันออก" อาจเป็นดังนี้คือ "ทุกๆ ค่าในรายการของอินพุทปรากฏอย่างแน่นอนเพียงครั้งเดียวในรายการเอาว์พุท รายการในเอาว์พุทมีค่ามาจากรายการอิน

พุทเท่านั้น และค่าในรายการเอาว์พุทมีอันดับเพิ่มขึ้น" คำอธิบายแบบวิธีดำเนินการกำหนดฟังก์ชันโดยให้อะกอริทึมของการคำนวณฟังก์ชันนั้น จุดประสงค์ของอะกอริทึมคือกำหนดว่าฟังก์ชันทำอะไรเท่านั้น ในระหว่างการสร้างระบบอะกอริทึมอื่นๆ ซึ่งคำนวณค่าเดียวกันสามารถนำมาใช้แทนกันได้ คำอธิบายแบบบอกเล่าดีกว่าคำอธิบายแบบวิธีดำเนินการ เพราะว่าคำอธิบายแบบบอกเล่าไม่ได้ส่อให้เห็นวิธีการสร้างระบบ แต่คำอธิบายแบบวิธีดำเนินการเขียนได้ง่ายกว่า ซึ่งควรจะเลือกใช้วิธีนี้

4.5.3 การกำหนดข้อบังคับระหว่างออบเจค

กำหนดข้อบังคับระหว่างออบเจค ข้อบังคับคือความไม่เป็นอิสระแห่งการกระทำระหว่างออบเจคซึ่งไม่ได้เกี่ยวพันกันเนื่องจากความไม่เป็นอิสระของค่าอินพุทและค่าเอาว์พุท ข้อบังคับสามารถอยู่บนออบเจคสองออบเจคที่เวลาเดียวกัน, ระหว่างสมาชิกของออบเจคเดียวกันในเวลาต่างกัน, หรือระหว่างสมาชิกของออบเจคที่แตกต่างกันในเวลาต่างกัน (สิ่งที่ตามมาภายหลังโดยปกติคือการกระทำเกี่ยวกับการอินพุทและเอาว์พุท) เงื่อนไขก่อนหน้าการกระทำคือข้อบังคับข้อมูลอินพุท และเงื่อนไขหลังการกระทำคือข้อบังคับข้อมูลเอาว์พุท

กำหนดค่าที่เป็นค่าสูงสุด, ค่าต่ำสุด, หรือมิฉะนั้นค่าที่ดี ถ้ามีมาตรฐานที่ดีหลายอย่างซึ่งขัดแย้งกันระบุว่าประโยชน์อย่างไรที่ต้องการ อย่ากังวลเกี่ยวกับการทำสิ่งนี้ให้ถูกต้องมากจนเกินไปนัก ปกติเราจะไม่สามารถทำได้ และมาตรฐานอาจเป็นไปได้ว่าเปลี่ยนก่อนที่จะทำโครงการเสร็จ

4.5.4 การเพิ่มโอเปอเรชัน

วิธีของการวิเคราะห์เชิงวัตถุไม่เน้นการกำหนดโอเปอเรชัน รายการของโอเปอเรชันที่มีประโยชน์ที่อาจเป็นไปได้คือเริ่มต้นและจบ มันยากที่จะรู้ว่าเมื่อไร ที่จะหยุดเพิ่มโอเปอเรชัน โอเปอเรชันในภาษาการเขียนโปรแกรมเชิงวัตถุตรงกับคำถาม (query) เกี่ยวกับแอทริบิวท์หรือแอสโซซิเอชันในออบเจคโมเดล อีเวนท์ในไดนามิกโมเดล และโพรเซสในฟังก์ชันนัลโมเดล เราพบว่ามีประโยชน์มากที่จะแยกชนิดความแตกต่างของโอเปอเรชันนัลในระหว่างการวิเคราะห์ ในภาษาเชิงวัตถุบางภาษา เช่น Smalltalk โอเปอเรชันเหล่านี้ทั้งหมดจะถูกพัฒนาขึ้นมาแบบเดียวกันแต่ภาษาอื่นๆ เช่น DSM แต่ละโอเปอเรชันจะไม่เหมือนกัน โอเปอเรชันสามารถเพิ่มได้จาก

- **ออบเจคโมเดล**

โอเปอเรชันจากโครงสร้างออบเจคมี การอ่าน, การเขียนค่าแอทริบิวท์ และแอสโซซิเอชันลิงค์ โอเปอเรชันเหล่านี้ไม่ต้องการถูกแสดงบนออบเจคโมเดล แต่เห็นได้จากการแทนค่าของแอทริบิวท์ ในระหว่างการวิเคราะห์สมมุติว่าแอทริบิวท์ทั้งหมดเข้าถึงได้ ใช้สัญญลักษณ์ "จุด" แสดงการเข้าถึงของแอทริบิวท์เช่น ATM.cash โมเดลสามารถถูกแสดงเป็นชุดของการเข้าถึง "pseudo-attribute" ได้เช่น account.bank การเข้าถึงควอลลิไฟลิงค์(qualified link) ได้โดยใช้สัญญลักษณ์ "ดัชนี" เช่น consortium.bank[bank-code].account[account-code] สัญญลักษณ์นี้สามารถถูกใช้ในซุโดโคดเพื่อกำหนดฟังก์ชันและแอคชันได้

- **อีเวนท์**

แต่ละอีเวนท์ที่ส่งถึงออบเจคตรงกับโอเปอเรชันบนออบเจคขึ้นกับสถาปัตยกรรมระบบอีเวนท์สามารถถูกสร้างขึ้นโดยตรงโดยจับอีเวนท์เป็นส่วนหนึ่งของพื้นฐานระบบ, หรือสามารถเปลี่ยนไปเป็นเมทธอด (method) ซึ่งมีอยู่ชัดเจนแล้ว ในระหว่างการวิเคราะห์อีเวนท์เขียนกำกับบนการเปลี่ยนสเตท และไม่ต้องเขียนในออบเจคโมเดล

- **สเตท แอคชัน และแอคทิวิตี**

แอคชันและแอคทิวิตีในสเตทไดอะแกรมอาจเป็นฟังก์ชัน ฟังก์ชันเหล่านี้มีโครงสร้างแห่งการคำนวณที่น่าสนใจควรถูกกำหนดเป็นโอเปอเรชันบนออบเจคโมเดล

- **ฟังก์ชัน**

แต่ละฟังก์ชันในดาค้าโฟลไดอะแกรมตรงกับโอเปอเรชันบนออบเจค (หรือเป็นไปได้ว่าหลายออบเจค)ฟังก์ชันเหล่านี้มีโครงสร้างแห่งการคำนวณที่น่าสนใจและจะสรุปบนออบเจคโมเดลจัดฟังก์ชันลงในโอเปอเรชันบนออบเจค

ถ้าชุดของสมการหรือส่วนหนึ่งของซูโดโคด อธิบายฟังก์ชันได้มากกว่าหนึ่งฟังก์ชัน ดังนั้นสามารถสร้างโอเปอเรชันใหม่เพื่อทำให้ฟังก์ชันนัลโมเดลง่ายขึ้น

- **ชอปปิ้งลิสท์โอเปอเรชัน(Shopping List Operation)**

บางครั้งพฤติกรรมของคลาสในความเป็นจริงเสนอแนะโอเปอเรชัน เมเยอร์ [Meyer-88] เรียกว่า "ชอปปิ้งลิสท์" เพราะว่าโอเปอเรชันไม่ได้ขึ้นกับทั้งระบบงานใดระบบงานหนึ่งและเรื่องลำดับใดลำดับหนึ่งของการทำให้สำเร็จ แต่มีความหมายในตัวเอง ชอปปิ้งลิสท์โอเปอเรชันอนุญาตให้คิดถึงความต้องการที่เป็นไปได้ในอนาคตขณะที่การจัดของคลาสยังคงเป็นแผนผังอยู่ ชอปปิ้งลิสท์โอเปอเรชันเตรียมโอกาสให้ฐานของคำนิยามออบเจคกว้างออก พ้นจากความต้องการของปัญหาที่คับแคบทันทีทันใด

ตรวจสอบออบเจคโมเดลสำหรับโอเปอเรชันที่คล้ายๆกันและการแปรผันรูปแบบบนโอเปอเรชันเดี่ยว พยายามทำให้คำนิยามของโอเปอเรชันกว้างออกเพื่อครอบคลุมถึงการแผรผันและกรณีพิเศษ ใช้อินแฮริทันซ์ในที่ๆ เป็นไปได้เพื่อลดจำนวนของโอเปอเรชันที่แตกต่างกัน สร้างซุปเปอร์คลาสใหม่เมื่อต้องการทำให้โอเปอเรชันง่ายขึ้น, ซึ่งการสร้างซุปเปอร์ใหม่นี้จะต้องไม่เป็นการบีบบังคับและผิดธรรมดา กำหนดที่ตั้งของแต่ละโอเปอเรชันในระดับที่ถูกต้องภายในแผนผังคลาสผลลัพธ์ของการขัดเกลานี้มักจะน้อยมาก โอเปอเรชันมีประสิทธิภาพมากกว่าและกำหนดได้ง่ายกว่าโอเปอเรชันเริ่มแรก เพราะว่ามันมีรูปแบบเดียวกันหมดและเป็นแบบธรรมดามากกว่า

โมเดลการวิเคราะห์ส่วนมากต้องการการวิเคราะห์มากกว่าหนึ่งครั้งที่จะสมบูรณ์ ประโยคปัญหาส่วนมากเป็นแบบวนรอบ และระบบงานส่วนมากไม่สามารถจะทำแบบเป็นเส้นตรง (linear) ได้เพราะว่าส่วนต่างๆของปัญหากระทำโต้ตอบกัน เพื่อที่จะเข้าใจปัญหาความเกี่ยวพันของมันทั้งหมดเราต้องทำวิเคราะห์อย่างซ้ำๆ เพื่อเพิ่มความเข้าใจของเรา มันไม่มีเขตที่แน่นอนระหว่างการวิเคราะห์และการออกแบบ ดังนั้นอย่าทำให้มันมากจนเกินไป ตรวจสอบการวิเคราะห์สุดท้ายกับเจ้าของระบบงาน และผู้เชี่ยวชาญระบบงาน

โมเดลการวิเคราะห์ทั้งหมดอาจแสดงความไม่คงเส้นคงวาและความไม่สมดุลภายในโมเดลและข้ามโมเดล ทวนสเตทต่างๆเพื่อไม่ให้มีสิ่งอื่นๆที่ไม่ต้องการ,ดูความต่อเนื่องกันในการออกแบบ พยายามขัดเกลาคำจำกัดความของออบเจคเพื่อการใช้ร่วมกัน และทำโครงสร้างให้ดีขึ้นเพิ่มรายละเอียดที่เราข้ามไปในตอนที่ทำครั้งแรก

บางโครงสร้างจะดูเหมือนมีสิ่งที่เกินความจำเป็นน่าที่จะปรับให้รัดกุมกว่านี้ ตรวจสอบใหม่ด้วยความระมัดระวังเราอาจจะมีแนวความคิดที่ผิดไป ในบางครั้งมีความต้องการจัดโครงสร้างหลักใหม่อีกครั้งในโมเดลเมื่อความเข้าใจของเราเพิ่มขึ้น มันทำได้ง่ายกว่าตอนที่ยังไม่เคยทำ ดังนั้นอย่าหลีกเลี่ยงที่จะเปลี่ยนที่ดูเหมือนเพิ่งจะถูกต้องเพียงเพราะว่าเรามีโมเดลเรียบร้อยแล้ว เมื่อมีโครงสร้างหลายโครงสร้างที่คล้ายๆ กันแต่เข้ากันไม่ได้เราอาจจะขาดแนวความคิดแบบธรรมดาหรือผิดกรณี ตัดส่วนประกอบของเจนเนอรัลไลเซชันบนแอทริบิวท์ที่ผิดออก

โดยปกติมักจะลืมออบเจคทางกายภาพ ซึ่งมีลักษณะทางลอจิคัลที่แตกต่างกันสองลักษณะ ซึ่งควรจะจำลองกับออบเจคที่แตกต่างกันสำหรับแต่ละลักษณะ การบ่งของสิ่งนี้คือออบเจคที่ไม่เหมาะเพราะมันต้องการใส่สองบทบาท

การบ่งบอกอื่นๆที่จะต้องดูคือข้อยกเว้นที่รวมอยู่, กรณีพิเศษหลายกรณีขาดความสมดุลที่คาดไว้ และออบเจคที่มีชุดของแอทริบิวท์หรือโอเปอเรชันที่ไม่เกี่ยวพันกันสองชุดหรือมาก

กว่า ลองคิดถึงการจัดโครงสร้างโมเดลใหม่อีกครั้ง เพื่อให้เก็บข้อบังคับภายในโครงสร้างได้ดีขึ้น ลบออบเจคหรือแอสโซซิเอชันที่ดูเหมือนมีประโยชน์ในครั้งแรกแต่ตอนนี้พบว่าเป็นของที่อื่น บ่อยครั้งที่ออบเจคสองออบเจคที่แตกต่างกันในการวิเคราะห์สามารถนำมารวมกันได้เพราะว่าความแตกต่างระหว่างออบเจคนั้นไม่ได้มีอิทธิพลเหนือส่วนที่เหลือของโมเดลในทางที่มีความหมายใดๆ

โมเดลที่ดีจะรู้สึกว่าถูกต้องและไม่ปรากฏว่ามีรายละเอียดของผู้อื่น อย่ากังวลถ้ามันดูเหมือนว่าไม่สมบูรณ์ แม้แต่โมเดลที่ดีและมีขนาดเล็กๆ ซึ่งการออกแบบเพียงพอแล้วก็ยังรู้สึกว่าไม่ถูกต้องเสียทีเดียว

เมื่อการวิเคราะห์สมบูรณ์ โมเดลเหมือนเป็นพื้นฐานสำหรับความต้องการและกำหนดขอบเขตของการสนทนาในอนาคต ความต้องการแท้ๆส่วนมากจะเป็นส่วนหนึ่งของโมเดล ความต้องการบางอย่างกำหนดข้อบังคับของการกระทำ สิ่งเหล่านี้ต้องถูกกำหนดอย่างชัดเจนร่วมกับมาตรฐานที่ดี ความต้องการอื่นๆ อาจจะกำหนดวิธีการแก้ปัญหา

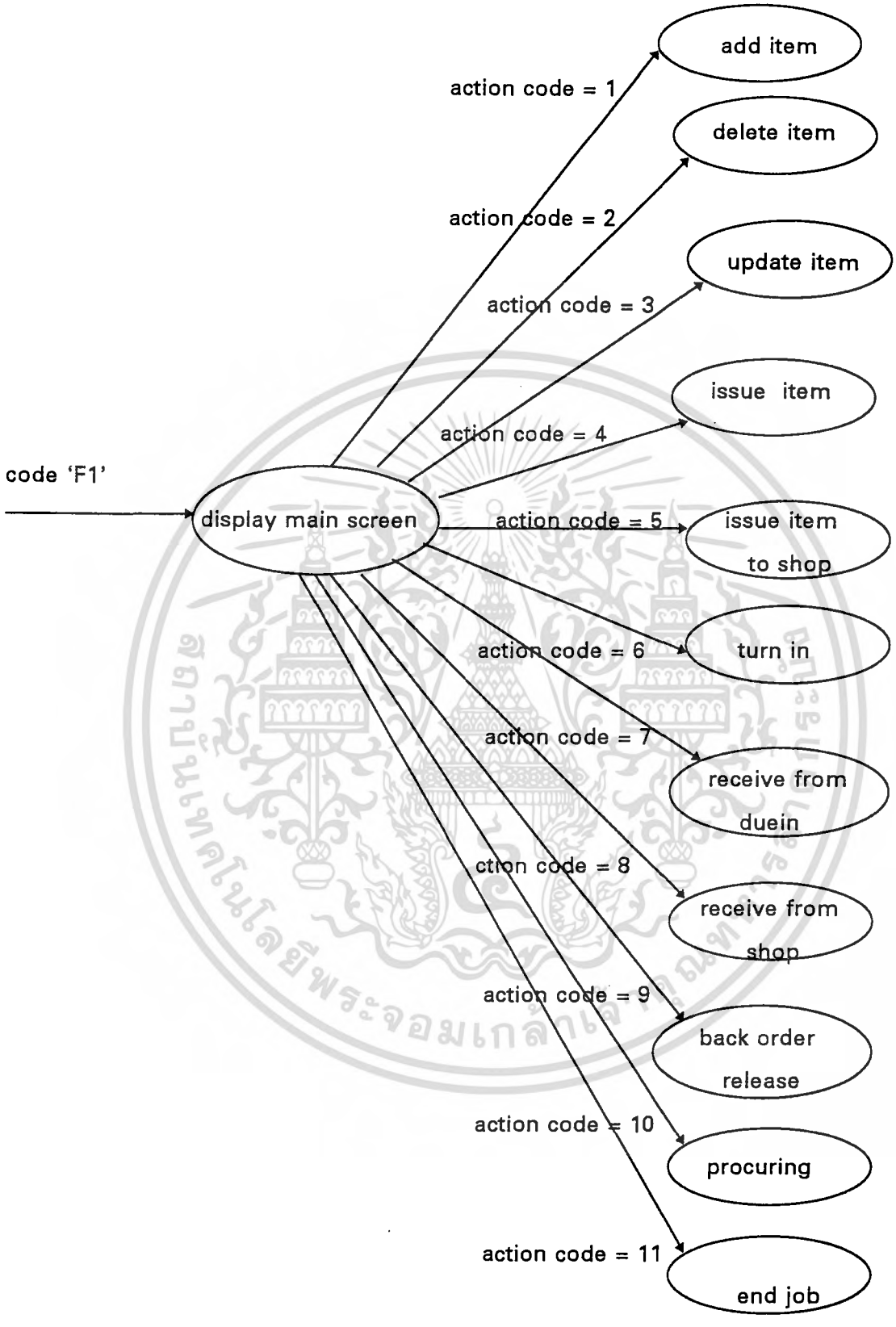
โมเดลขั้นสุดท้ายต้องถูกตรวจสอบกับเจ้าของระบบงาน ในระหว่างการวิเคราะห์ความต้องการบางอย่างอาจปรากฏว่าไม่ถูกต้องหรือทำไม่ได้ การแก้ไขความต้องการต้องได้รับการยืนยัน โมเดลการวิเคราะห์ต้องถูกตรวจสอบโดยผู้เชี่ยวชาญระบบงานด้วยเพื่อสร้างความมั่นใจว่าโมเดลถูกต้องและเป็นจริง เราพบว่าโมเดลการวิเคราะห์คือวิธีของการติดต่อสื่อสารกับผู้เชี่ยวชาญระบบงานที่ไม่ใช่ผู้เชี่ยวชาญคอมพิวเตอร์ที่ได้ผล

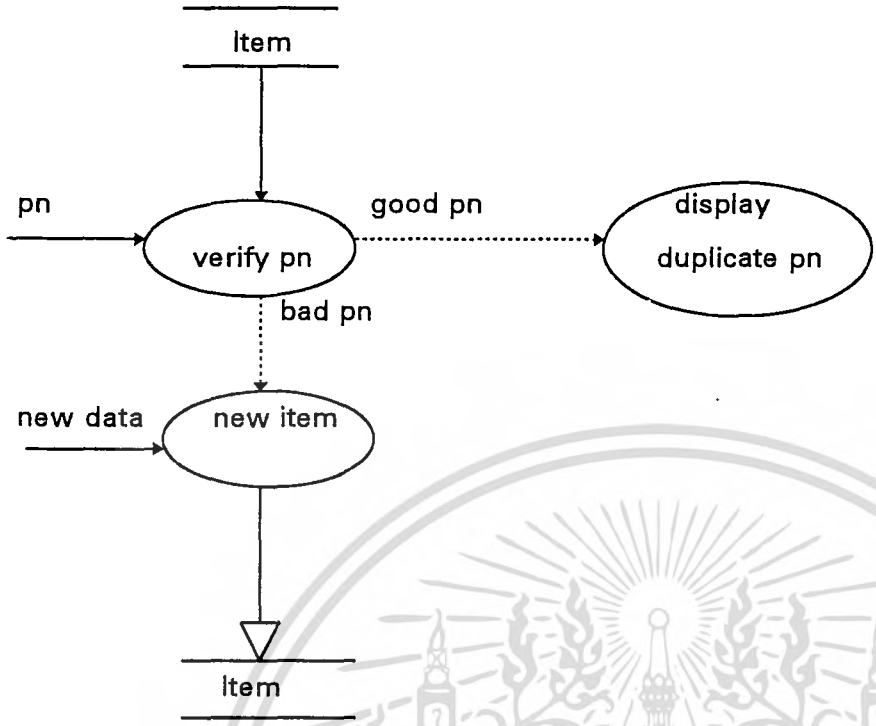
โมเดลการวิเคราะห์ที่ถูกตรวจสอบขั้นสุดท้ายเพื่อเป็นพื้นฐานสำหรับสถานปัตยกรรมระบบ การออกแบบ และการสร้างระบบ ประโยคปัญหาเริ่มแรกต้องถูกทวนเพื่อรวมความถูกต้องเข้าด้วยกันกับความเข้าใจที่พบระหว่างการวิเคราะห์

เป้าหมายของการวิเคราะห์คือกำหนดปัญหาและขอบเขตระบบงานอย่างครบถ้วนโดยไม่มีการพยายามสร้างให้สอดคล้องกับการสร้างระบบแต่อย่างใด แต่ในการปฏิบัติมันอาจจะเป็นไปไม่ได้ที่จะหลีกเลี่ยงไม่ให้มีสิ่งที่เกี่ยวกับการสร้างระบบเลย ไม่มีเส้นแบ่งที่แน่นอนในแต่ละตอนที่ออกแบบว่าขั้นตอนนี้คือการวิเคราะห์ พ้นเขตแดนนี้ไปเป็นออกแบบระบบ เวลาปฏิบัติจริงจะมีการคาบเกี่ยวกัน ไม่ได้เป็นไปตามกฎที่กล่าวมาข้างต้นอย่างเข้มงวดแต่สามารถยืดหยุ่นได้ เป้าหมายของการจำลองแบบคือทำงานทั้งหมดให้สำเร็จ

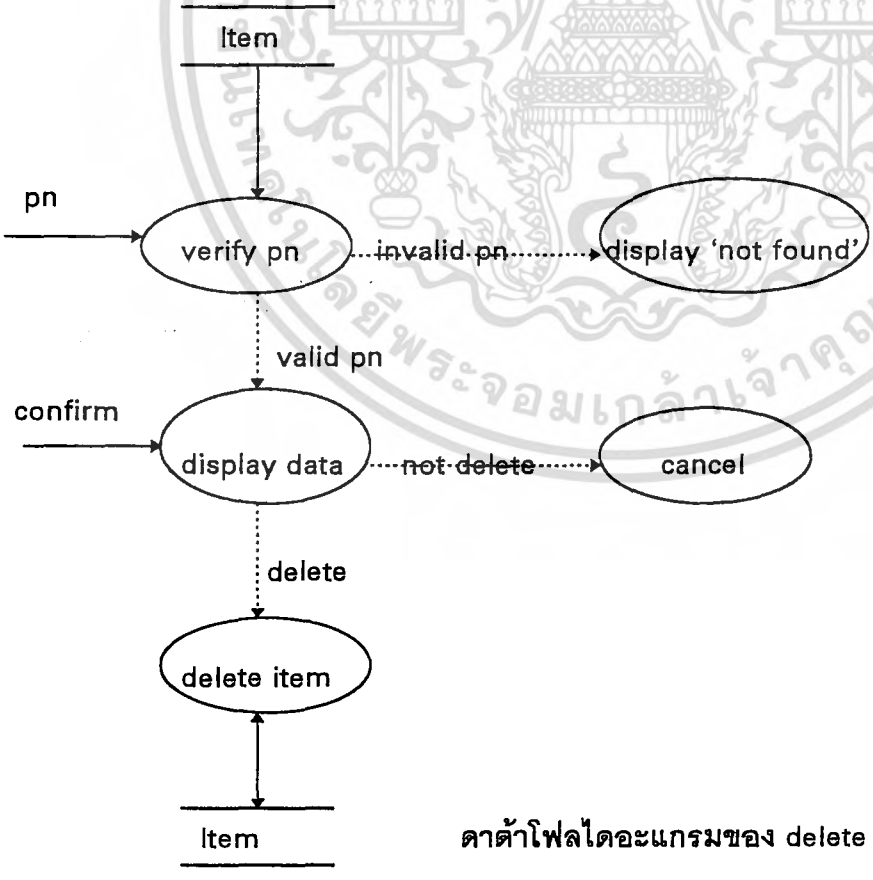
รายละเอียดฟังก์ชันนัลโมเดลของระบบงานบริหารพัสดุสายช่างอากาศแสดงไว้ในภาพที่ 36

ภาพที่ 36 ฟังก์ชันนัลโมเดล

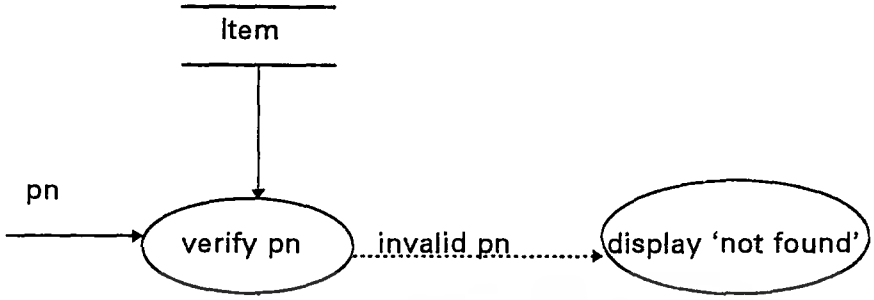


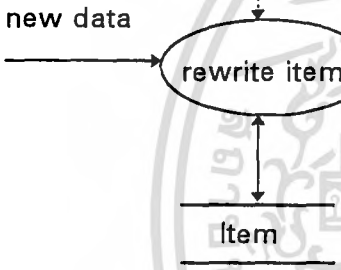


ดาต้าโฟลไดอะแกรมของ add item process

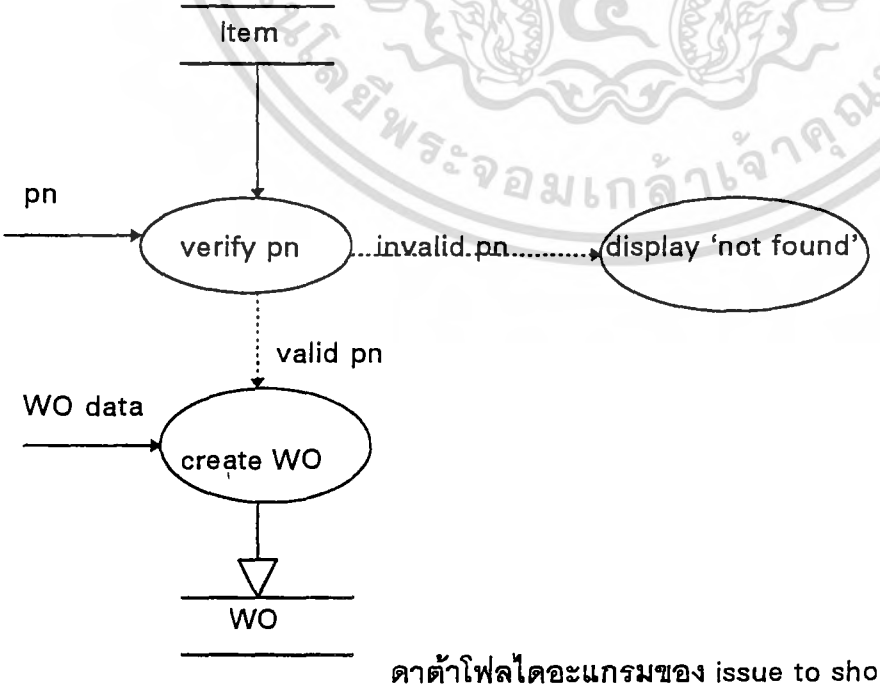


ดาต้าโฟลไดอะแกรมของ delete item process

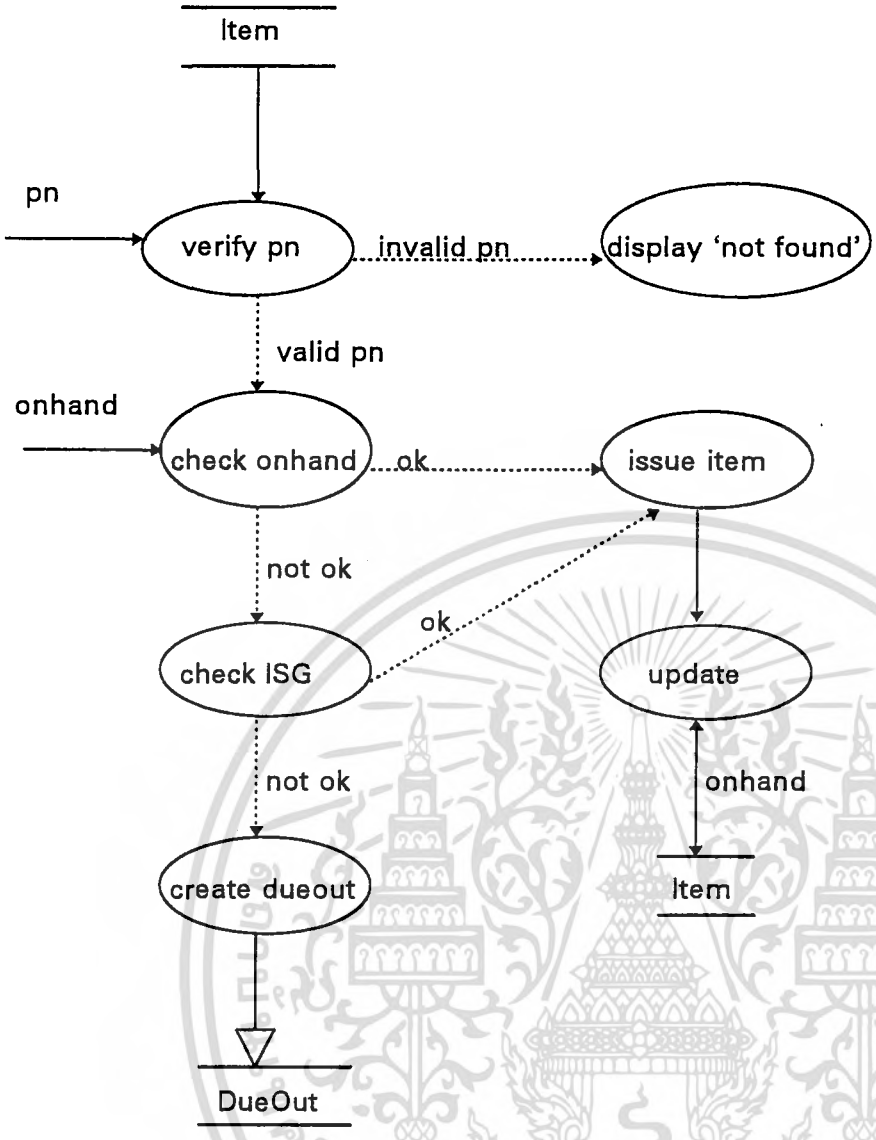




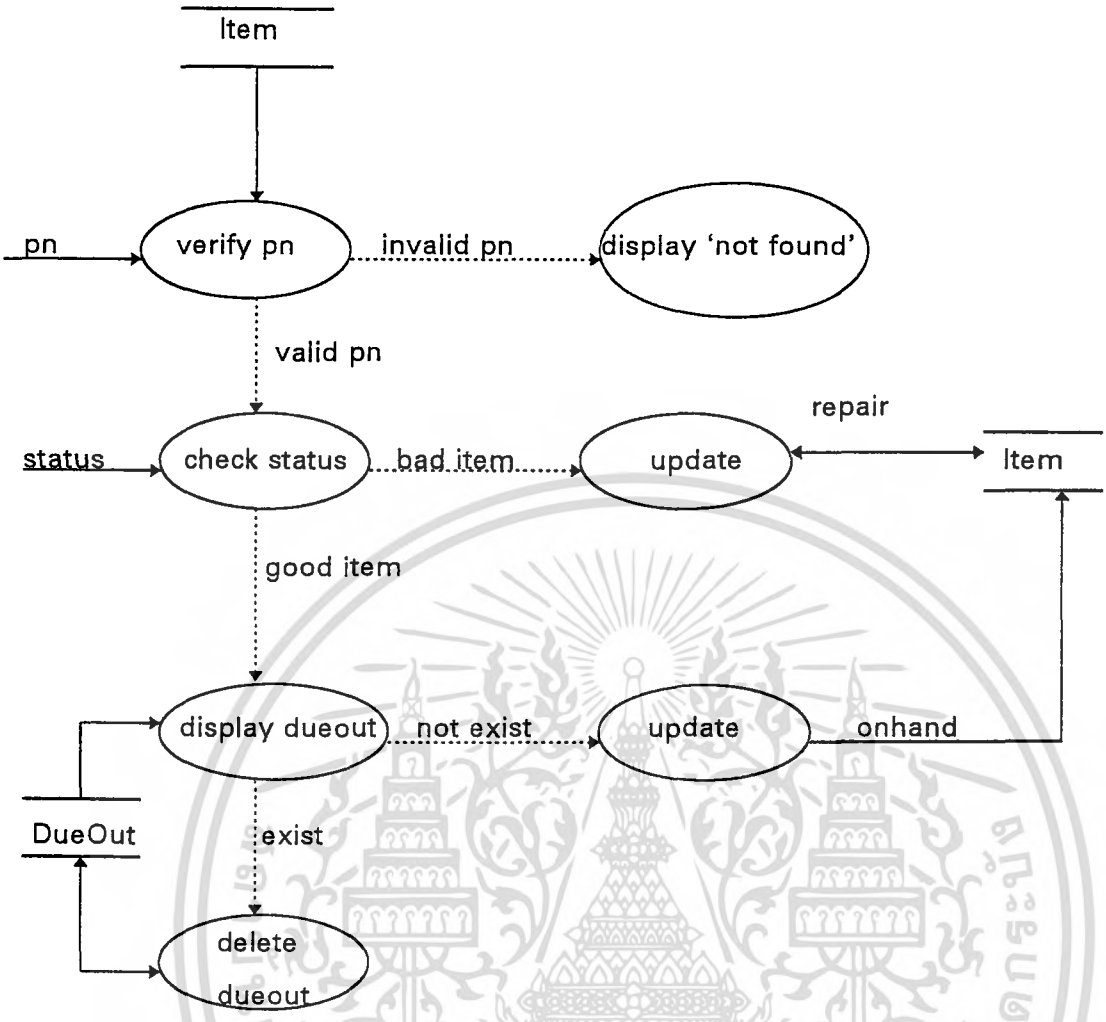
ดาต้าโฟลไดอะแกรมของ update item process



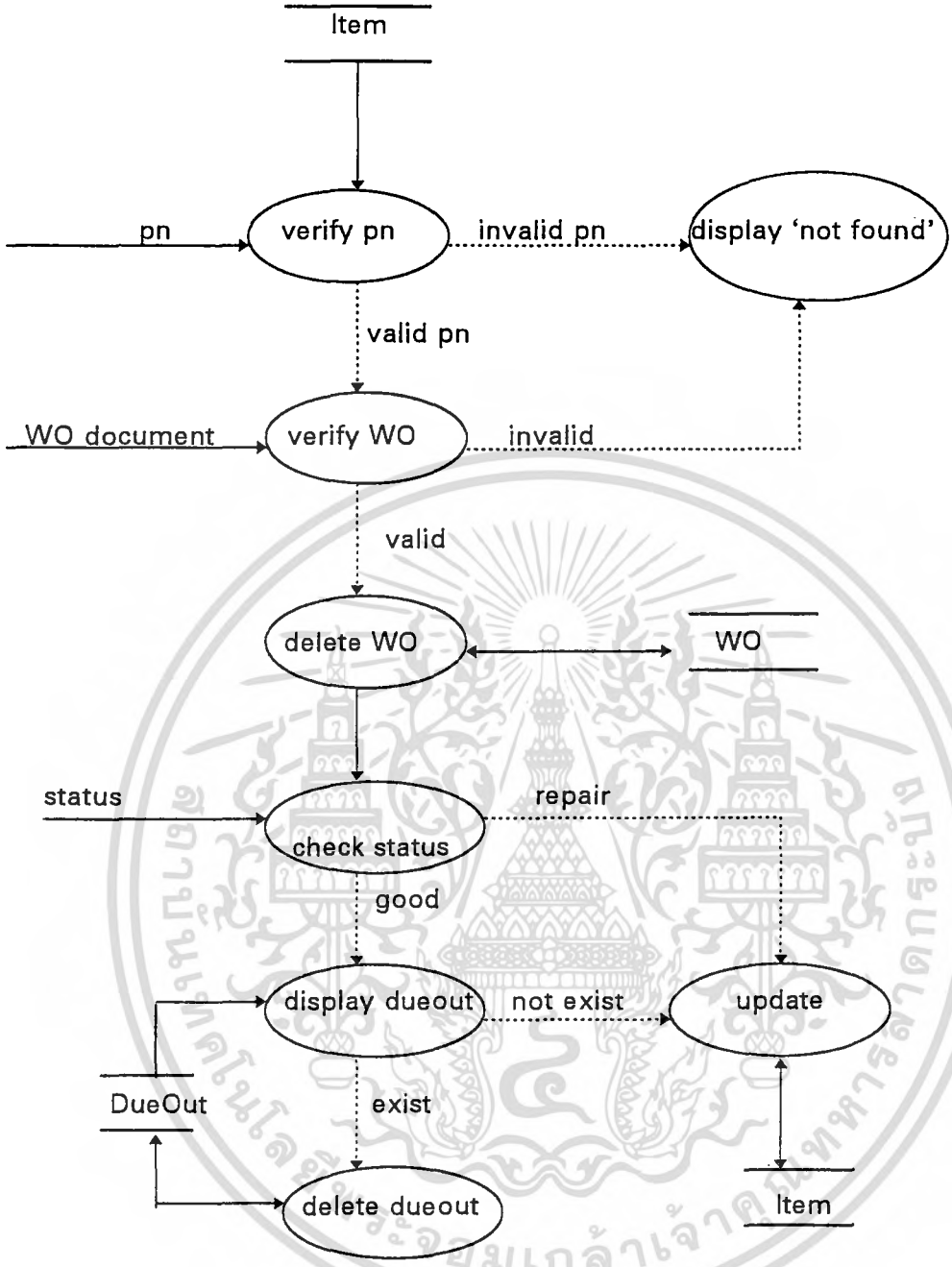
ดาต้าโฟลไดอะแกรมของ issue to shop process



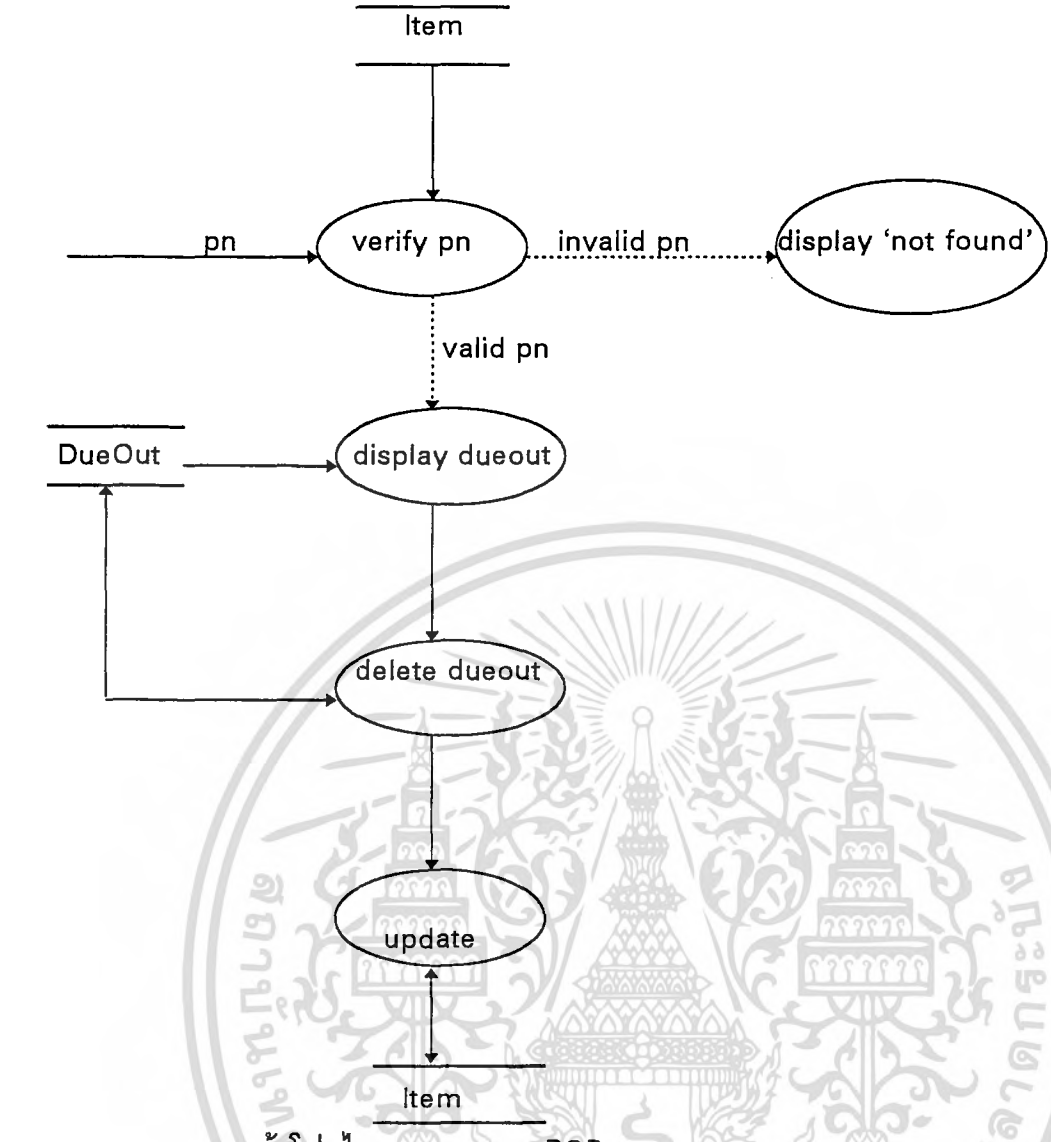
ดาต้าโฟลไดอะแกรมของ issue item process



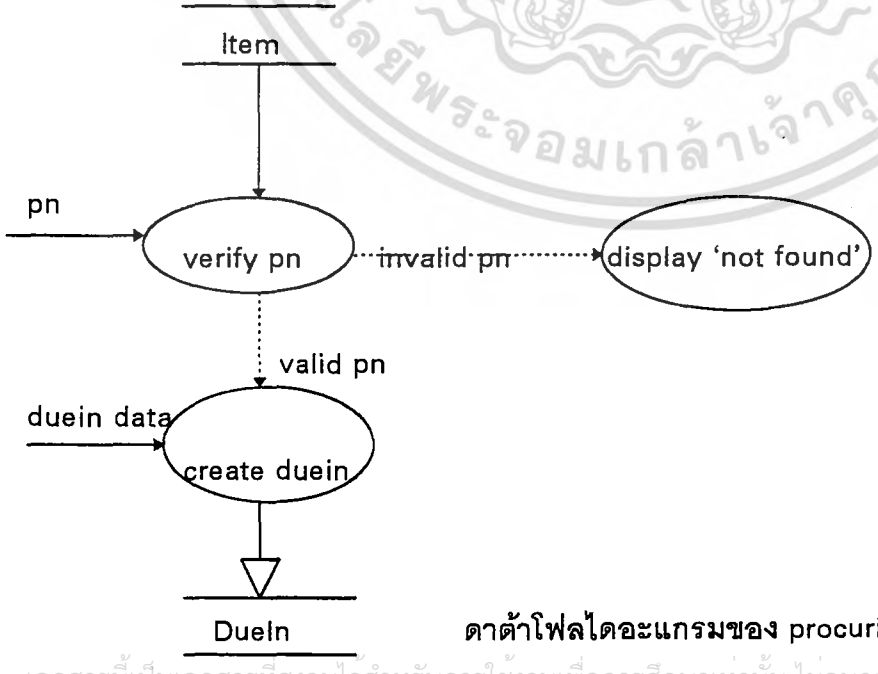
ดาต้าโฟลไดอะแกรมของ turn in process



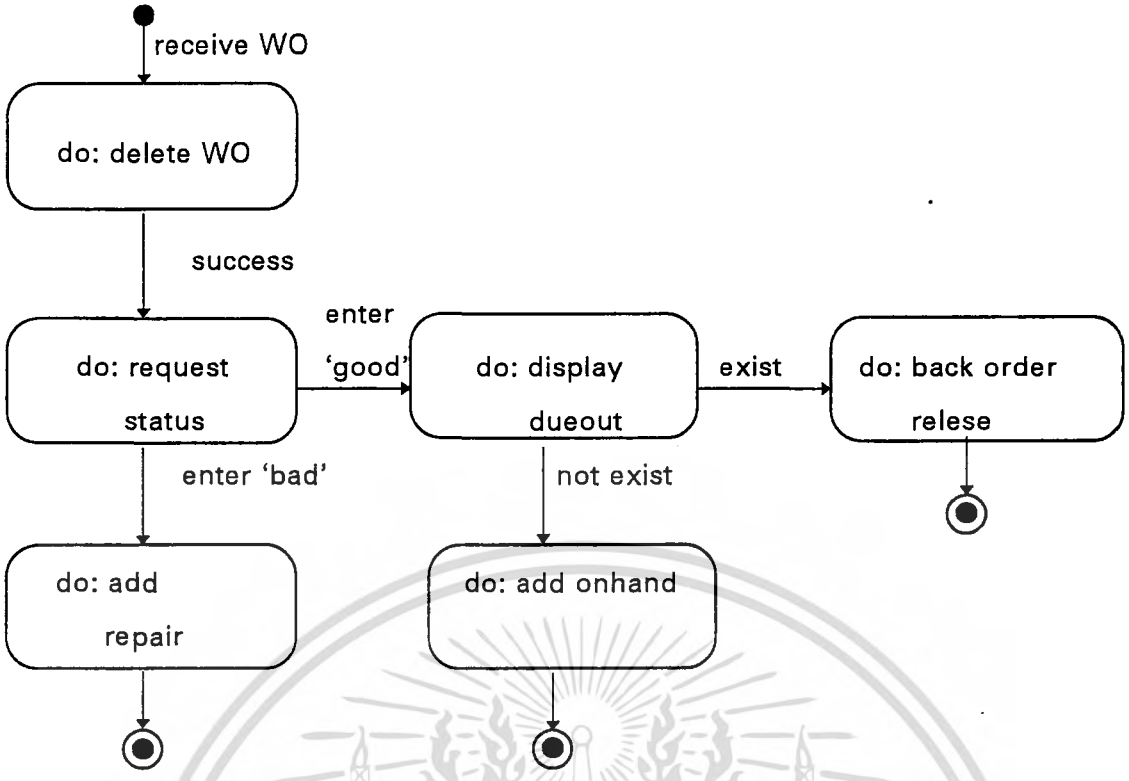
ดาต้าโฟลไดอะแกรมของ receive from shop process



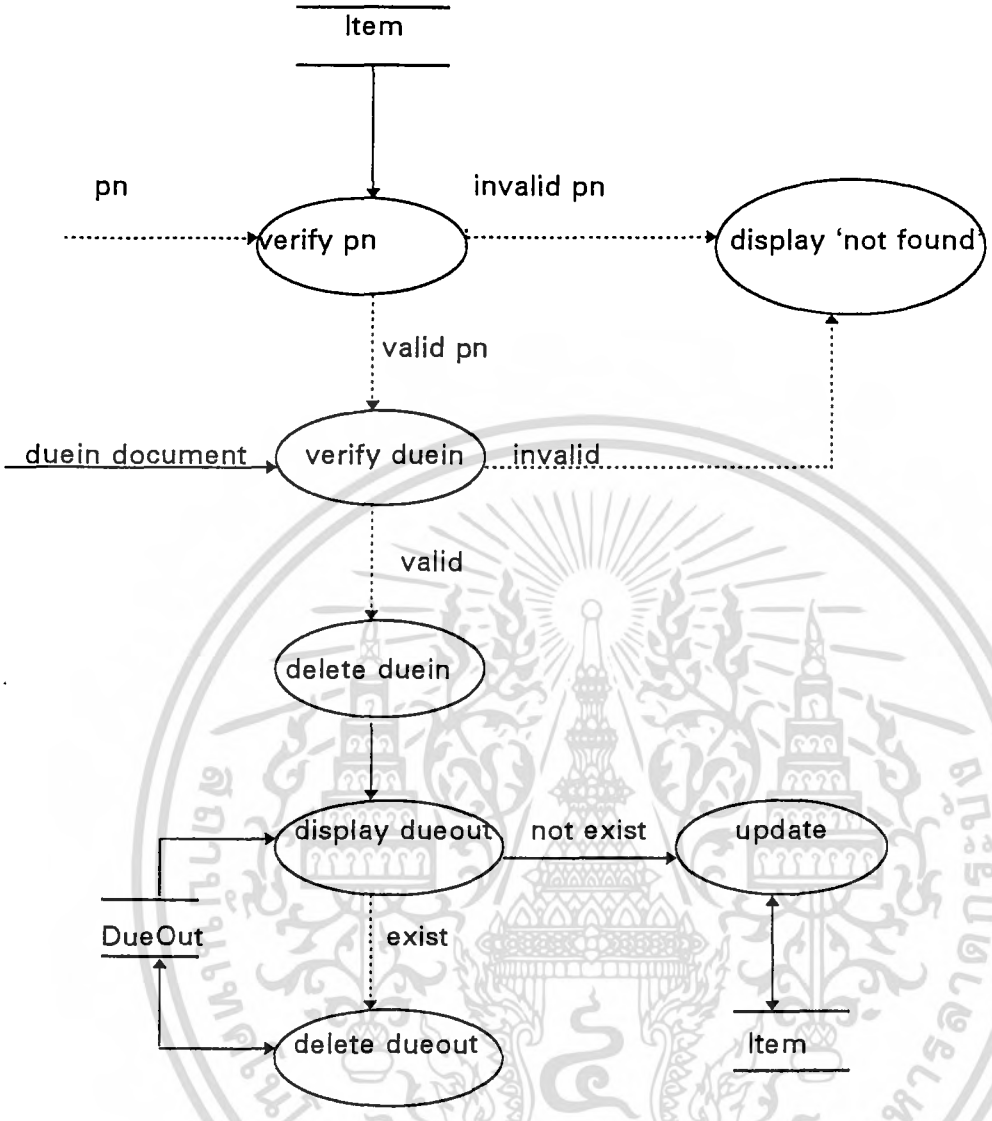
ดาต้าโฟลไดอะแกรมของ BOR



ดาต้าโฟลไดอะแกรมของ procuring process



สเตทไดอะแกรมของคลาส Item



ดาต้าโฟลไดอะแกรมของ receive from duein process

บทที่ 5
การบำรุงรักษาระบบ

5.1 การบำรุงรักษา

สิ่งที่จะต้องทำในเรื่องของการบำรุงรักษาคือ

1) จะเก็บรักษาไฟล์ของระบบอย่างไรถ้าขนาดของไฟล์ใหญ่ขึ้น

2) ขั้นตอนปกติที่เราสามารถป้องกันไม่ให้งานสูญหายไป ถ้าหากว่าระบบเกิดมีปัญหาทำงานไม่ได้

3) จะทำอย่างไรถ้าระบบเกิดปัญหาทำงานไม่ได้

5.2 การบันทึกการเปลี่ยนแปลงแบบอัตโนมัติ

สมอลทอคจะบันทึกการเปลี่ยนแปลงไว้ในไฟล์เชนจ์ล็อก (Change Log) ส่วนโคดที่ไม่มีการเปลี่ยนแปลงจะเก็บไว้ในไฟล์ซอส (Source) และเก็บพอนย์เตอร์ของไฟล์เชนจ์ล็อก ไฟล์ซอสไว้ในไฟล์อิมเมจ (Image) ดังนั้นเราต้องเก็บรักษาไฟล์ทั้งสามไว้ด้วยกัน จากเหตุผลดังกล่าวเราจึงไม่ควรที่จะปรับปรุงแก้ไขไฟล์เชนจ์ล็อก เราสามารถเรียกมาดูได้และสามารถที่จะนำเมธธอดและการกำหนดคลาสกลับเข้ามาในระบบได้อีกในกรณีที่มีการกู้ข้อมูล (Recovery) อันเนื่องมาจากระบบเกิดปัญหาทำงานไม่ได้ เหตุการณ์ที่สำคัญอื่นๆ ที่ทำให้สมอลทอคมีการจดบันทึกแบบอัตโนมัติ เช่น ทุกครั้งที่สั่งสำเนาไฟล์อิมเมจ (ใช้คำสั่ง Save Image) จะบันทึกข้อความพร้อมวันเวลาลงในไฟล์เชนจ์ล็อก ทุกครั้งที่สั่งให้มีการทำงาน (ใช้คำสั่ง do it หรือ show it) จะมีการจดบันทึกรวมทั้งเมื่อมีการสั่งให้ลบเมธธอดออกจากคลาสก็จะมีการจดบันทึกเช่นกัน

5.3 ความสำคัญของการเก็บอิมเมจ

ไฟล์อิมเมจคือกลุ่มของออบเจคทุกออบเจคในสมอลทอคซึ่งถูกสร้างขึ้นมา เมื่อเราสั่งสำเนาข้อมูลในไฟล์อิมเมจคำอธิบายของออบเจคทั้งหมดจะบันทึกลงในไฟล์อิมเมจ เมื่อเราเริ่มต้นระบบงานใหม่ในภายหลังทุกๆ ออบเจคในไฟล์อิมเมจจะถูกสร้างขึ้นมาอีกครั้งหนึ่งเฉพาะตรงจุดที่ไฟล์อิมเมจถูกสั่งให้ทำสำเนาไว้ก่อนที่จะออกจากระบบงานครั้งล่าสุด ดังนั้นระบบจะเริ่มต้นใหม่เหมือนกับที่เราได้เคยใช้งานระบบก่อนหน้านั้น เมื่อเราเปลี่ยนแปลงโคด (Code) หรือสร้างออบเจคใหม่และเปลี่ยนแปลงรายละเอียดของออบเจคเดิมเราจะต้องสั่งทำสำเนาไฟล์อิมเมจเพื่อ

ให้การเปลี่ยนแปลงนั้นถาวร เราสามารถสั่งสำเนาไฟล์อิมเมจด้วยการเลือกใช้ฟังก์ชันเซฟอิมเมจในเมนูระบบ หรือเลือกฟังก์ชันเซฟอิมเมจเมื่อตอนที่จะออกจากโปรแกรมสมอลทอค ถ้าเราไม่ได้สั่งให้ทำสำเนาไฟล์อิมเมจไว้เราจะไม่ได้การเปลี่ยนแปลงต่างๆ เหล่านั้นเมื่อเราเริ่มต้นระบบใหม่ในคราวหน้า

5.4 การลดขนาดไฟล์เชนจ์ล็อก (Compressing Change Log)

การลดขนาดของไฟล์เชนจ์ล็อกนั้นจะมีการสร้างสำเนาของเมธธอดล่าสุดของแต่ละเมธธอดที่เป็นเมธธอดใหม่หรือมีการเปลี่ยนแปลง ไฟล์อิมเมจใหม่จะถูกเขียนขึ้นโดยอัตโนมัติเราควรลดขนาดของไฟล์เชนจ์ล็อกเมื่อมันเริ่มจะใหญ่ขึ้น ซึ่งในขณะที่จะทำการลดขนาดของไฟล์เชนจ์ล็อกนี้เนื้อที่ในดิสท์จะต้องมีพอสำหรับไฟล์เชนจ์ล็อกเก่าและใหม่ในเวลาเดียวกันด้วย ดังนั้นเราต้องมั่นใจว่าเราได้ลดขนาดของไฟล์เชนจ์ล็อกก่อนที่ดิสท์จะเต็ม คำสั่งที่ใช้คือ Smalltalk compressChanges

5.5 การลดขนาดของไฟล์ซอส (Compressing source)

การลดขนาดของไฟล์ซอสจะมีการสร้างซอสไฟล์ใหม่สำหรับทุกๆ เมธธอดในระบบในขณะนั้น โดยที่จะคัดลอกสำเนาล่าสุดของซอสโคดสำหรับแต่ละเมธธอดจากไฟล์ซอสเก่าหรือจากไฟล์เชนจ์ล็อก หลังจากการทำให้ขนาดของไฟล์ซอสลดลงแล้วจะได้ไฟล์อิมเมจและไฟล์เชนจ์ล็อกใหม่เกิดขึ้นโดยอัตโนมัติด้วย ไฟล์เชนจ์ล็อกใหม่ที่ได้จะเป็นไฟล์ที่ว่างเปล่า ไฟล์ซอสจะถูกเขียนในรูปแบบบีบเพื่อประหยัดเนื้อที่ในดิสท์ ดังนั้นเราไม่สามารถแก้ไขไฟล์ซอสได้นอกจากโปรแกรมสมอลทอคเอง แต่เราสั่งทำสำเนาไฟล์ซอสได้ หลังจากที่เราบีบซอสเราจะได้ไฟล์อิมเมจใหม่ ไฟล์ซอสใหม่ และไฟล์เชนจ์ล็อกใหม่ แต่ไฟล์เชนจ์ล็อกจะว่างเปล่า ดังนั้นเราจึงควรที่จะทำสำเนาของไฟล์ทั้งสามไว้ก่อนที่จะทำการลดขนาดของไฟล์ซอสเผื่อว่าอาจจะมีข้อผิดพลาดอะไรเกิดขึ้นจะได้สามารถกู้ข้อมูลคืนได้ คำสั่งที่ใช้คือ Smalltalk compressSources

5.6 การกู้ระบบ (Surviving a System Crash)

สมอลทอคมีความยืดหยุ่นกับข้อผิดพลาดมาก ซึ่งออกจะโชคไม่ดีนักที่ความเสียหายสามารถที่จะเกิดขึ้นได้โดยเฉพาะถ้าหากว่าเรามีการปรับปรุงแก้ไขในสมอลทอค แต่ว่าคุณสมบัติการบันทึกสำเนาแบบอัตโนมัติของสมอลทอคทำให้การกู้ข้อมูลสามารถทำได้ ความเสียหายไม่

ได้เกิดขึ้นเพราะว่าในโปรแกรมสมอลทอคมีบัก (bug) แต่เพราะว่ามีการเปลี่ยนแปลงสภาวะภายในของโปรแกรมสมอลทอคอันเนื่องมาจากโปรแกรมสมอลทอคยอมให้เราเปลี่ยนแปลงอะไรก็ได้ ทุกอย่างแม้แต่การทำลายสภาวะของระบบ แต่สิ่งเหล่านี้ก็ไม่ได้เป็นปัญหาอะไร เพราะว่าเราสามารถกู้ข้อมูลกลับคืนมาได้เกือบทั้งหมด ในความเป็นจริงการทดลองทำกับระบบคือวิธีการเรียนรู้ที่ดีที่สุด แต่ว่าเราควรจะมั่นใจว่าเราได้ทำสิ่งต่อไปนี้เป็นการป้องกันไว้ก่อนล่วงหน้าแล้วคือ

1) ต้องทำสำเนาของซอสไฟล์ไว้เสมอ

2) ต้องทำสำเนาที่เชื่อถือได้ของไฟล์อิมเมจและเซนจ์ล็อก จะต้องจำไว้ว่าไฟล์อิมเมจและไฟล์เซนจ์ล็อกใช้ในการทำงานร่วมกัน ซึ่งถ้าหากว่าเราใช้ไฟล์อิมเมจที่เป็นเวอร์ชันเก่ากว่าเวอร์ชันของไฟล์เซนจ์ล็อกเราก็จะไม่สามารถใช้ซอสโคดสำหรับบางเมธธอดได้ เพราะว่าหลังจากที่เราลดขนาดของไฟล์เซนจ์ล็อกแล้วจะทำให้ไฟล์อิมเมจและไฟล์เซนจ์ล็อกจะถูกเปลี่ยนไปด้วย เราจึงไม่สามารถใช้ไฟล์อิมเมจเก่ากับไฟล์เซนจ์ล็อกใหม่ที่ถูกลดขนาดลงแล้วได้แต่เราจะใช้ไฟล์อิมเมจเก่ากับไฟล์เซนจ์ล็อกเวอร์ชันล่าสุดได้ถ้าหากว่าไฟล์เซนจ์ล็อกนั้นยังไม่ได้ถูกลดขนาดลงในช่วงระยะเวลาที่ไฟล์อิมเมจกำลังใช้ไฟล์เซนจ์ล็อก สรุปก็คือจะต้องทำสำเนาไฟล์อิมเมจและไฟล์เซนจ์ล็อกร่วมกันแล้วเราจะไม่มีปัญหา

3) เราควรจะสำเนาไฟล์อิมเมจเก็บไว้ในดิสท์ก่อนที่เราจะพยายามทำบางสิ่งบางอย่างที่อาจจะทำให้ระบบเกิดปัญหาจนไม่สามารถทำงานได้ ซึ่งเราสามารถที่จะใช้คำสั่งทำสำเนาไฟล์อิมเมจได้จากเมนูระบบ การทำเช่นนี้จะช่วยให้เราสามารถรักษางานทั้งหมดของเราไว้ได้ถ้าหากว่าระบบเกิดปัญหาจนทำงานไม่ได้ขึ้นมาเราก็จะสามารถเริ่มต้นใหม่อีกครั้งได้โดยการใช้สำเนาไฟล์อิมเมจนี้ แต่ถ้าหากว่าเราทำสำเนาไฟล์อิมเมจยังไม่สำเร็จและระบบเกิดมีปัญหาไม่ทำงานขึ้นมาเราก็ยังสามารถที่จะใช้ไฟล์เซนจ์ล็อกช่วยได้

4) ถ้าเราทดลองกับโปรแกรมสมอลทอคและเราเกิดไปทำระบบเสียหรือว่าเราไม่ต้องการให้การเปลี่ยนแปลงนั้นถาวรเราก็ไม่ต้องทำไฟล์สำเนาอิมเมจ โดยการที่เราออกจากโปรแกรมสมอลทอคด้วยการใช้ฟังก์ชันฟอร์เกทอิมเมจ (forget image) แต่ว่าการเปลี่ยนแปลงทั้งหมดเหล่านั้นถูกบันทึกไว้ในไฟล์เซนจ์ล็อกแล้วและเราก็สามารถที่จะเรียกกลับเข้ามาในระบบอีกครั้งได้

5) ถ้าระบบเกิดมีปัญหาไม่ทำงาน อย่าตกใจ ให้ทำสำเนาของไฟล์อิมเมจและไฟล์เซนจ์ล็อก โดยปกติเราสามารถที่จะกู้ข้อมูลกลับคืนมาได้ เพราะถ้าเราทำสำเนาเอาไว้เราก็สามารถที่จะค่อยๆ อ่านไปทีละส่วนเพื่อหาข่าวสารที่ต้องการจะกู้ข้อมูลคืนมา

5.7 การกู้ข้อมูล (Recovering From a System Crash)

ถ้าระบบไม่ทำงานเราจะต้องทำสำเนาของไฟล์อิมเมจและไฟล์เซนจ์ล็อกไว้ก่อนแล้ว การกู้ข้อมูลมีลำดับขั้นตอนดังนี้

1) ตรวจสอบถ้าหากว่าเรามีไฟล์อิมเมจที่ดีแล้ว พยายามที่จะเริ่มต้นโปรแกรมสมอลทอคแบบปกติโดยการใช้ไฟล์อิมเมจนี้ แต่ถ้ามันเริ่มต้นไม่ได้เราควรจะหาสำเนาของไฟล์อิมเมจและไฟล์เซนจ์ล็อกที่ใหม่ที่สุดแล้วเริ่มต้นโปรแกรมสมอลทอคด้วยเวอร์ชันนั้น

2) ในขณะที่โปรแกรมสมอลทอคกำลังทำงาน ให้มองหาในไฟล์เซนจ์ล็อกที่กำลังใช้ในขณะนั้นเมื่อระบบเกิดไม่ทำงาน ในไฟล์เซนจ์ล็อกจะมีการเปลี่ยนแปลงทั้งหมดที่เราทำและการทำงานทั้งหมดที่เราสั่ง

3) มองหาจุดที่เราทำสำเนาไฟล์อิมเมจล่าสุดในไฟล์เซนจ์ล็อกเพราะว่าทุกครั้งที่เราทำสำเนาไฟล์อิมเมจโปรแกรมสมอลทอคจะเขียนข้อความบอกวันเวลาที่ทำในไฟล์เซนจ์ล็อก ไล่หาในไฟล์เซนจ์ล็อกจากข้างท้ายขึ้นมาถ้าต้องการสำเนาอิมเมจล่าสุด

4) เมื่อได้จุดที่ทำสำเนาอิมเมจซึ่งงานของเราได้หายไปที่จุดนี้ไปจนจบไฟล์เซนจ์ล็อก ใช้ฟังก์ชันอินสทอล (install) จากเมนูเพน (pane menu) เราต้องระวังที่จะไม่นำเอาส่วนที่มีข้อผิดพลาดกลับเข้ามาในระบบอีก

บทที่ 6
สรุปและข้อเสนอแนะ

6.1 สรุปผลการวิจัย

ผลการทำวิจัยสรุปได้ดังนี้

1. การศึกษาทฤษฎี

การออกแบบระบบโดยใช้หลักการเชิงวัตถุแตกต่างไปจากการออกแบบระบบงานแบบเดิม ซึ่งใช้วิธีการเก็บรวบรวมข้อมูลต่างๆ เป็นแฟ้มข้อมูล(File) หรือฐานข้อมูล(Database) แยกกันกับโปรแกรมระบบงาน ในขณะที่หลักการเชิงวัตถุจะกำหนดสิ่งต่างๆที่อยู่ภายในขอบเขตของระบบงานเป็นออบเจค และในออบเจคจะประกอบด้วยข้อมูลและเมธธอด รวมกันอยู่ภายในออบเจค การเรียกใช้เมธธอดหรือการสั่งให้เกิดการกระทำตามที่กำหนดไว้ในเมธธอดคือการส่งแมสเซสไปยังออบเจคนั้นๆ ดังนั้นหากมีการเปลี่ยนแปลงวิธีการกระทำใดๆของออบเจคจะไม่มีผลกระทบกับข้อความที่ส่งไปยังออบเจคนั้นและจะไม่มีผลกระทบกับออบเจคอื่นๆด้วย นั่นคือโปรแกรมของระบบไม่มีการเปลี่ยนแปลง ซึ่งนับว่าเป็นผลดีอย่างมากกับระบบงานเพราะโดยปกติแล้วพบว่าระบบงานมักจะมีการเปลี่ยนแปลงวิธีการปฏิบัติอยู่เสมอๆ เมธธอดสามารถใช้อธิบายถึงพฤติการณ์ของออบเจคแต่ละออบเจคได้ ออบเจคที่มีพฤติการณ์แบบเดียวกันสามารถจัดรวมกลุ่มเป็นกลุ่มเดียวกันได้เรียกว่าคลาส ในระหว่างคลาสสามารถที่จะถ่ายทอดคุณสมบัติ จากคลาส หนึ่งไปยังอีกคลาสหนึ่งได้ โดยเรียกคลาสที่ถ่ายทอดคุณสมบัติว่าซุปเปอร์คลาส และเรียกคลาสที่ได้รับการถ่ายทอดคุณสมบัติว่าซับคลาส ซึ่งทำให้การพัฒนาระบบงานทำได้ง่าย ในการทำวิจัยนี้เลือกใช้โมเดลของออบเจคโมเดลลิ่งเทคนิค(Object Modeling Technique) หรือโอเอ็มที(OMT) และเลือกใช้โปรแกรมภาษาสมอลทอคเพราะเป็นต้นแบบของภาษาเชิงวัตถุโดยแท้จริง

2. การวิเคราะห์เชิงวัตถุ(Object-Oriented Analysis)

ระบบงานบริหารพัสดุสายช่างอากาศ กำหนดคลาส Item เป็นคลาสของพัสดุทั้งหมด คลาส Item ประกอบด้วยซับคลาส 2 ซับคลาส คือคลาส Reparable และคลาส Consumable โดยคลาส Reparable คือคลาสของพัสดุประเภทพัสดุซ่อมหมุนเวียนและคลาส Consumable คือคลาสของพัสดุประเภทพัสดุสิ้นเปลือง หลังจากนั้นสร้างแบบจำลองเพื่อใช้อธิบายระบบงานทั้งระบบงานซึ่งมีอยู่ด้วยกัน 3 แบบจำลองคือหนึ่งออบเจคโมเดล (Object Model) ใช้อธิบายรายละเอียดเกี่ยวกับว่าออบเจคนั้นว่ามีข้อมูลและเมธธอดอะไรบ้าง และแสดงความสัมพันธ์ระหว่างออบเจคแต่ละ

ออบเจคภายในระบบงาน แบบจำลองที่สองคือไดนามิกโมเดล (Dynamic Model) ซึ่งใช้สเตทไดอะแกรม (State Diagram) อธิบายพฤติกรรมของออบเจคที่กระทำโต้ตอบกับอีเวนท์ต่างๆ ไดนามิกโมเดลนี้ไม่ค่อยมีความจำเป็นมากนักสำหรับระบบงานบริหารพัสดุสายช่างอากาศ เพราะว่าเป็นระบบงานที่เกี่ยวกับการเก็บข้อมูลเสียเป็นส่วนใหญ่ และแบบจำลองสุดท้ายคือฟังก์ชั่นนัลโมเดล (Functional Model) ซึ่งใช้ดาต้าโฟลไดอะแกรม (Data Flow Diagram) อธิบายถึงวิธีการคำนวณค่าของเมธธอดภายในออบเจคแต่ละออบเจค

3. การพัฒนาระบบงานด้วยโปรแกรมสมอลทอค

จากผลของการทำวิจัยเริ่มตั้งแต่การออกแบบระบบงานไปจนกระทั่งถึงการพัฒนาระบบงานด้วยโปรแกรมสมอลทอค พบว่าสามารถที่จะวิเคราะห์และออกแบบระบบงานได้ดีเช่นเดียวกับวิธีการแบบเดิมและจะดีกว่า,ออกแบบได้ง่ายกว่าสำหรับระบบงานบางประเภทเสียด้วยซ้ำเช่น ระบบงานที่มีข้อมูลซับซ้อนมากๆ หรือระบบงานที่มีการเปลี่ยนแปลงวิธีการทำ (procedure) บ่อยๆ ระบบงานที่ได้จากการวิจัยสามารถทำงานได้ดีเช่นเดียวกับวิธีการแบบเดิม ในด้านของผู้ใช้ก็ไม่รู้สึกว่ามีอะไรเปลี่ยนแปลงเปรียบเสมือนการสั่งดอกไม้ ผู้ขายจะใช้วิธีการใดลูกค้าไม่รู้ รู้แต่ว่าได้รับดอกไม้ตามที่ต้องการ ตรงกันข้ามกับวิธีการแบบเดิมที่ส่วนมากแล้วเมื่อมีการแก้ไขผู้ใช้ต้องมีการอบรมใหม่ ในระยะแรกๆ ของการทำอาจจะรู้สึกติดขัดบ้างเพราะว่ายังไม่คุ้นกับแนวการคิดแบบใหม่ แต่เมื่อเริ่มคุ้นเคยจะพบว่าสามารถทำได้ง่ายและสะดวกกว่าวิธีการแบบเดิม การสร้างความคุ้นเคยกับแนวความคิดแบบใหม่นี้ก็ใช้เวลาไม่มากนักเพราะหลักการเข้ากับชีวิตประจำวันของเราที่มักจะมองสิ่งต่างๆ รอบตัวเราเป็นออบเจคๆ หนึ่ง

6.2 ข้อเสนอแนะ

งานวิจัยนี้มุ่งที่จะทดลองใช้หลักการเชิงวัตถุมาออกแบบระบบงานแทนวิธีการแบบเดิมๆ เพื่อจะศึกษาหาข้อสรุปผลการทำงาน ดังนั้นจึงเลือกใช้โปรแกรมสมอลทอค ซึ่งเป็นโปรแกรมเชิงวัตถุโดยแท้ แต่ว่าในปัจจุบันมีซอฟแวร์เชิงวัตถุใหม่ๆ เกิดขึ้นมากมาย ซึ่งใช้งานได้ง่ายและสะดวกกว่า จึงขอเสนอแนะให้ผู้ที่สนใจได้ทดลองใช้โปแกรมเชิงวัตถุใหม่ๆ เหล่านั้น

ภาคผนวก ก
รายละเอียดบทความทางวิชาการที่ตีพิมพ์

ปีที่ 20 ฉบับที่ 105 / มกราคม - กุมภาพันธ์ 2537

คอมพิวเตอร์สาร

JOURNAL OF COMPUTER ASSOCIATION OF THAILAND UNDER THE ROYAL PATRONAGE OF HIS MAJESTY THE KING

การเชื่อมโยงสื่อสารระยะไกล กับเครื่องแปลภาษา

คอมพิวเตอร์กับจิตสำนึก

กรณีออกแบบฐานข้อมูลข้าราชการและพลเรือน
- ระบบเครือข่ายฯ ของข้าราชการแห่งอนาคต
- กรมอู่ทหารเรือ

การใช้โปรแกรมกับงานบริหารข้าราชการทหารอากาศ

นำกฎของ Fuzzy มาใช้ปรับปรุงภาพ

แบบแห่งการจำลองสีและแสง

ดร.สุกมีศร จิตตะยโสธร
น.ส.หญิง นพพร สงวนทรัพย์

การใช้โปรแกรมเชิงวัตถุกับระบบงานบริหารพัสดุสายช่างอากาศ

ในบทความนี้ได้นำเสนอแนวความคิดของการเขียนโปรแกรมเชิงวัตถุ (Object-Orienterd Programming) ซึ่งกำหนดข้อมูลประเภทนามธรรม (Abstraction Data Type หรือ ADT) ออบเจ็กต์ (Object) สร้างมาจากเอดีที (ADT) ออบเจ็กต์ประกอบด้วยตัวแปรที่มีฟังก์ชันที่จำเป็นที่กระทำกับตัวแปรเหล่านั้นรวมอยู่ด้วย ตัวแปรใช้แทนข้อมูลข่าวสารภายใน ฟังก์ชันใช้กำหนดการกระทำที่กระทำกับออบเจ็กต์นั้นออบเจ็กต์ใหม่สามารถถ่ายทอด (Inheritance) คุณสมบัติจากออบเจ็กต์ที่มีอยู่เดิมได้ คำสั่งเดียวกันให้ผลการทำงานต่างกันในออบเจ็กต์ที่ต่างกัน (Polymorphism) การเปลี่ยนแปลงข้อมูลภายในออบเจ็กต์ไม่ทำให้โปรแกรมที่เรียกใช้ฟังก์ชันต้องเปลี่ยนแปลง ทำให้การบำรุงรักษาและการขยายโปรแกรมทำได้ง่าย ซึ่งเหมาะสมกับข้อมูลที่มีความซับซ้อนและระบบที่มีขนาดใหญ่ ดังนั้นจึงได้นำเอาหลักการนี้มาใช้ออกแบบระบบงานบริหารพัสดุสายช่างอากาศ ซึ่งประสบปัญหาในด้านการเปลี่ยนแปลงโครงสร้างข้อมูลและต้องแก้ไขโปรแกรมด้วยเสมอ ทำให้มีปัญหาในการปรับปรุงระบบงาน

Oabject-Oriented Programming (OOP) relies on three basic concepts:data abstraction, inheritance, and polymorphism. Data abstraction refers to ability to define abstract data type or ADT that encapsulate some data together with a set to well-defined operations. Such user-defined data types preaetined hierarchy with a view to share can represent in software. Inheritance is the mechanism that enables one object to behave just like another, except for some modifications. Polymorphism refers to the fact the different objects react differently to the same message. The aeronautical supply management system organizes programs using a collection of objects whose classes are organized in code and data through inheritance.

1. บทนำ

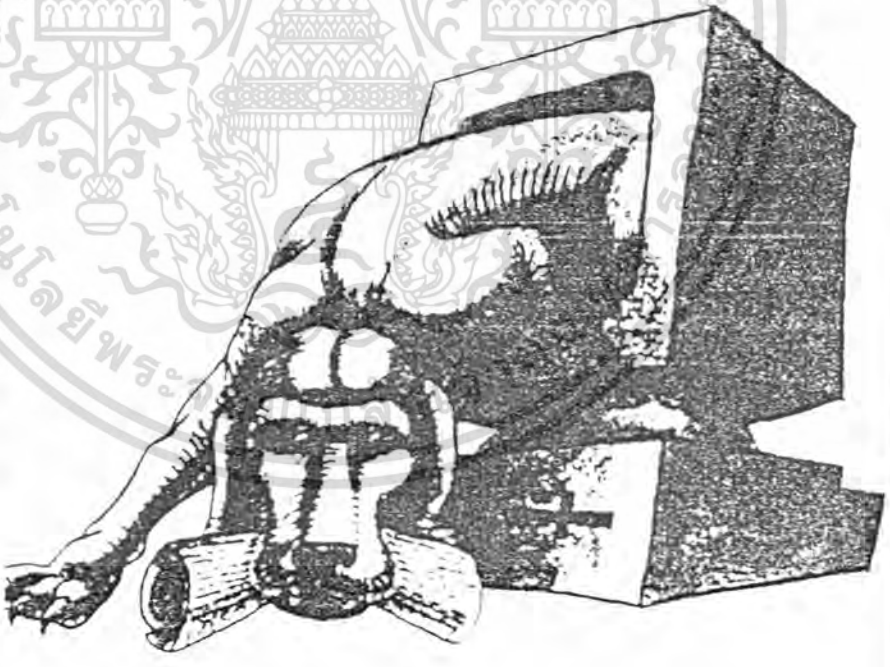
ตั้งแต่คอมพิวเตอร์ยุคแรกโปรแกรมเมอร์พยายามหาวิธีการเขียนโปรแกรมที่มีความซับซ้อนโปรแกรมคอมพิวเตอร์ในระยะแรกคือ ลำดับของคำสั่งภาษาเครื่องที่ถูกนำ (Fetch) เข้าไปในหน่วยความจำโดยหน่วยประมวลผลกลางและทำงานตามคำสั่งนั้น

ภาษาแอสเซมบลีได้ปรับปรุงให้สามารถใช้ชื่อนีโมนิก (mnemonic) แทนคำสั่งภาษาเครื่อง และใช้ชื่อของสัญลักษณ์แทนตำแหน่งของหน่วยความจำได้ ตัวแปลที่เรียกว่าแอสเซมเบลอร์จะเปลี่ยนโปรแกรมภาษาแอสเซมบลีให้เป็นภาษาเครื่อง ในเวลาต่อมาแอสเซมเบลอร์เริ่มมีคำสั่งเฉพาะหรือไดเรกทีฟ (directive) ซึ่งยอมให้โปรแกรมเมอร์จัดกลุ่มของข้อมูลพื้นฐานเป็นโครงสร้างของข้อมูลแล้วตั้งชื่อโครงสร้างของข้อมูลนั้น และเขียนโปรแกรมโครงสร้างของข้อมูลนั้น และเขียนโปรแกรม โครงสร้าง แต่แอสเซมเบลอร์ยังคงบังคับให้แนวความคิดในการเขียนโปรแกรมอยู่ในรูปแบบของคำสั่งภาษาเครื่อง ซึ่งซีพียูสามารถทำงานตามคำสั่งนี้ได้

ภาษาในระดับที่สูงกว่า เช่น ภาษาฟอร์แทรน เบสิก ปาสกาล และภาษา C ได้ขจัดการผูกติดกับคำสั่งภาษาเครื่องของซีพียู โดยการกำหนดประเภทของข้อมูลมาตราฐานเช่นเลขจำนวนเต็ม เลขจำนวนทศนิยม และตัวอักษร ฯลฯ ซึ่งสามารถใช้ในประโยคคำสั่งที่ถูกแปลให้เป็นคำสั่งภาษาเครื่อง โปรแกรมใช้งานที่มีอยู่ส่วนมากเขียนขึ้นในรูปแบบผสมกันระหว่างภาษาระดับสูงกับภาษาแอสเซมบลี รูปแบบการเขียนโปรแกรมมาที่ได้รับการยอมรับมีการจัดข้อมูลที่มีความสัมพันธ์กันโดยใช้คำสั่งโครงสร้าง เช่น คำสั่ง Record ในภาษาปาสกาล หรือคำสั่ง Struct ในภาษา C แล้วจัดกลุ่มของข้อมูลที่ได้ให้เป็นหน่วยเดียว เมื่อมีการจัดโครงสร้างของข้อมูลดังนี้การเขียนโปรแกรมใช้งานจึงเขียนในรูปของชุดคำสั่งซึ่งกระทำกับโครงสร้างของข้อมูลเรียกชุดคำสั่งนี้ว่าโพรซีเยอร์ (Procedure)

ถึงแม้ว่าการเขียนโปรแกรมด้วยวิธี "ออกแบบโครงสร้างของข้อมูลและเขียนฟังก์ชันที่กระทำกับข้อมูล" สามารถเขียนได้ดี แต่ความซับซ้อนของซอฟต์แวร์ได้เพิ่มมากขึ้นเพื่อให้ฮาร์ดแวร์มีขีดความสามารถเพิ่มขึ้นเช่นซีพียูเร็วขึ้น ทำกราฟิกให้ดีขึ้นทำระบบเครือข่ายง่ายขึ้น ฯลฯ ผู้ใช้ยอมคาดหวังว่าความสามารถของฟังก์ชันจะต้องเพิ่มขึ้น และโปรแกรมจะรวมคุณลักษณะต่าง ๆ เอาไว้ด้วย เช่น ระบบการติดต่อของผู้ใช้กับระบบวินโดว์ ระบบการบันทึกข้อมูลในคอมพิวเตอร์ทั้งขนาดเล็กและขนาดใหญ่ และความสามารถทำงานในสภาวะของระบบเครือข่าย ฯลฯ จากสาเหตุของความซับซ้อนเหล่านี้ทำให้มีโปรแกรมเมอร์หลายคนหันมาใช้เทคนิคของการเขียนโปรแกรมเชิงวัตถุ (Object-Oriented Programmimg หรือ OOP)

ไม่ได้มีอะไรเป็นสิ่งใหม่ในการเขียนโปรแกรมเชิงวัตถุ แนวความคิดก็คือการกำหนดข้อมูลชนิดนามธรรม (data abstaction) การถ่ายทอด (inheritance) และการใช้คำสั่งเดียวกันแต่ให้ผลการทำงานที่แตกต่างกัน (polymorphism) แล้วอะไรคือความใหม่ที่



เป็นสิ่งที่เพิ่มความสนใจในหมู่ของโปรแกรมเมอร์ทั่วไปและโปรแกรมเมอร์ภาษา C เหตุผลข้อหนึ่งก็คือความนิยมการเขียนโปรแกรมภาษา C++ ซึ่งโปรแกรมเมอร์หลายคนมองเหมือนว่าเป็นส่วนที่พัฒนาต่อมาจากภาษา C และเป็นภาษาที่ในปัจจุบันผู้พัฒนาซอฟต์แวร์ส่วนมากเลือกใช้ ภาษา C++ ได้เพิ่มคำสั่งโครงสร้างเข้าไปในภาษา C หลายอย่างซึ่งช่วยให้ทำเทคนิคเชิงวัตถุได้โดยตรง ถ้าเป็นโปรแกรมเมอร์ภาษา C อยู่แล้วจะเรียนรู้เกี่ยวกับกฎเกณฑ์ของภาษา C++ ได้ง่ายขึ้น เพียงแต่ต้องเปลี่ยนแปลงแนวความคิดเสียใหม่ถ้าต้องการจะใช้หลักการเขียนโปรแกรมเชิงวัตถุ วิธีที่ดีที่สุดในการเรียนรู้ภาษา C++ คือการทำความเข้าใจกับแนวความคิดของโอโอพี (OOP) แล้วดูว่าภาษา C++ ทำตามแนวความคิดนั้นได้อย่างไร

2. อะไรคือการเขียนโปรแกรมเชิงวัตถุ

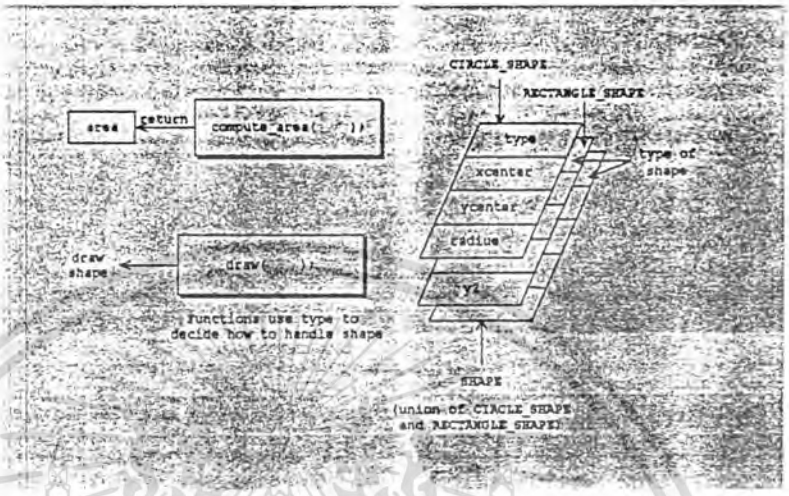
การเขียนโปรแกรมเชิงวัตถุได้นิยามข้อมูลชนิดนามธรรมเพื่อใช้แทนข้อมูลจริงที่มีความซับซ้อน หรือใช้แทนออบเจ็กต์ชนิดนามธรรมแล้วสร้างโปรแกรมโดยใช้ข้อมูลแอบสแทรคชัน (Data abstraction) กล่าวถึงการใช้นิยามของข้อมูลชนิดนามธรรม ส่วนในหัวข้อของ Inheritance และ Polymorphism นั้นกล่าวถึงกลไกที่ทำให้สามารถได้รับประโยชน์จากคุณลักษณะของข้อมูลชนิดนามธรรมซึ่งก็คือ ออบเจ็กต์ในโอโอพีนั่นเอง

ก่อนที่จะเข้าไปสู่หัวข้อของโอโอพี พิจารณาข้อสังเกตสองข้อ ข้อหนึ่งโอโอพีคือวิธีการออกแบบซอฟต์แวร์และการทำให้ซอฟต์แวร์นั้นเป็นจริงขึ้นมาเท่านั้น การใช้เทคนิคของการเขียนโปรแกรมเชิงวัตถุไม่ได้บ่งบอกถึงความสำเร็จที่สามารถมองเห็นได้ของซอฟต์แวร์ อย่างไรก็ตามเมื่อโปรแกรมเมอร์จะทำซอฟต์แวร์ให้เป็นจริงขึ้นมาก็สามารถที่จะหวังประโยชน์จากการใช้วิธีการเขียนโปรแกรมเชิงวัตถุได้โดยเฉพาะซอฟต์แวร์ของโครงการขนาดใหญ่ เพราะว่าโอโอพีทำให้สามารถอยู่ในระดับแนวความคิด (Conceptual) ได้และสร้างแบบจำลองระดับสูงของปัญหาที่กำลังพยายามจะแก้ปัญหานั้นได้ซึ่งสามารถจัดการกับความซับซ้อนต่าง ๆ ได้ดีกว่าวิธีอื่น ๆ ที่มักจะบังคับให้จัดปัญหาให้เข้ากับลักษณะของภาษานั้น สามารถได้รับประโยชน์จากความเปลี่ยนแปลง ของออบเจ็กต์ และทำโปรแกรมให้เป็นจริงขึ้นมาในรูปของหน่วยที่มีความสัมพันธ์แบบเป็นอิสระไม่ขึ้นต่อกัน ซึ่งสามารถจัดการดูแลและขยายได้ง่ายโดยสามารถจะใช้รหัสในออบเจ็กต์ร่วมกันได้โดยวิธีการถ่ายทอด

ข้อสองคือโอโอพีไม่ได้ขึ้นกับภาษาโปรแกรมใด ๆ ถึงแม้ว่าภาษาโปรแกรมนั้นจะทำเทคนิคของโอโอพีได้ก็เพียงแต่ทำให้เทคนิคนั้นเป็นจริงขึ้นมาได้ง่ายขึ้นเท่านั้น

3. การเขียนโปรแกรมโครงสร้าง Procedure Oriented Programming)

ก่อนที่จะพูดถึงหัวข้อโอโอพี มาดูภาษาที่ใช้เขียนโปรแกรมโครงสร้าง เช่น ภาษา C วิธีเขียนโปรแกรมโครงสร้างคือจัดลำดับของปัญหาเป็นลำดับของขั้นตอนที่จะต้องทำ จัดข้อมูลที่มีความสัมพันธ์กันโดยใช้คำสั่งในภาษา C หรือ STRUCT เขียนฟังก์ชันหรือโพรซีเยอร์ที่จำเป็นกระทำกับข้อมูลและต้องใช้ในการทำงาน จัดลำดับของงานให้สมบูรณ์เพื่อใช้งานการแก้ปัญหา ถึงแม้ว่าข้อมูลจะถูกจัดเป็นแบบโครงสร้าง จุดสนใจจุดแรกอยู่ที่ฟังก์ชัน ตัวอย่างโปรแกรมภาษา C เช่น การจัดการเกี่ยวกับรูปทรงเรขาคณิต คือรูปสี่เหลี่ยมและวงกลม โปรแกรมต้องสามารถที่จะวาดรูปทรงใด ๆ และคำนวณพื้นที่ของรูปนั้นได้ นั่นคือสามารถแบ่งการทำงานของโปรแกรมได้เป็นสองฟังก์ชัน คือวาดรูปและคำนวณพื้นที่ เริ่มต้นกำหนดโครงสร้างข้อมูลทั้งสองรูปทรงแล้วเชื่อมให้เป็นหน่วยเดียวกันด้วยคำสั่ง UNION โดยมีรหัสกำหนดว่าขณะนั้นใช้โครงสร้างข้อมูล ของรูปทรงใด จากนั้นเขียนฟังก์ชันทั้งสองฟังก์ชันของแต่ละรูปทรงโดยใช้คำสั่ง SWITCH เป็นตัวระบุว่าจะใช้รูปทรงใดดังรูปที่ 1



รูปที่ 1 *การจัดการเกี่ยวกับรูปทรงเรขาคณิตด้วยภาษา C*

บัญหาข้อหนึ่งของการเขียนโปรแกรมโครงสร้างคือเมื่อต้องการเพิ่มโครงสร้างของข้อมูลแบบใหม่ เช่นรูปสามเหลี่ยม จะต้องทำดังต่อไปนี้

ขั้นที่ 1 กำหนดโครงสร้างของรูปสามเหลี่ยม

ขั้นที่ 2 เพิ่มสมาชิกใหม่ในส่วนของ UNION

ขั้นที่ 3 เพิ่มรูปทรงใหม่ในฟังก์ชันทั้งสอง

ขั้นที่ 4 แก้ไขตัวโปรแกรม เพิ่มส่วนที่ต้องการใช้โครงสร้างของข้อมูลที่เพิ่มใหม่

เหตุผลที่แสดงตัวอย่างข้างต้นเพื่อชี้ให้เห็นว่าเมื่อต้องการเพิ่มประเภทของข้อมูลแบบใหม่ต้องมีการแก้ไขตลอดทั้งโปรแกรม และในความเป็นจริงการเขียนโปรแกรมเกี่ยวข้องกับแฟ้มข้อมูลมากมายนั่นคือ เมื่อมีการเปลี่ยนแปลงจะต้องแก้ไขแฟ้มข้อมูลทุก ๆ แฟ้มข้อมูล

ในหัวข้อถัดไปจะเห็นว่าการใช้วิธีของโปรแกรมเชิงวัตถุสามารถหลีกเลี่ยงบัญหานี้ได้ โดยโครงสร้างของข้อมูลและฟังก์ชันที่กระทำกับข้อมูลนั้นเหมือนเดิม มีการแก้ไขโปรแกรมน้อยที่สุด นี่คือข้อดีของโอโอพี

4. คำศัพท์ของโอโอพี

มีแนวความคิด 3 ข้อภายใต้หัวข้อโอโอพี คือ

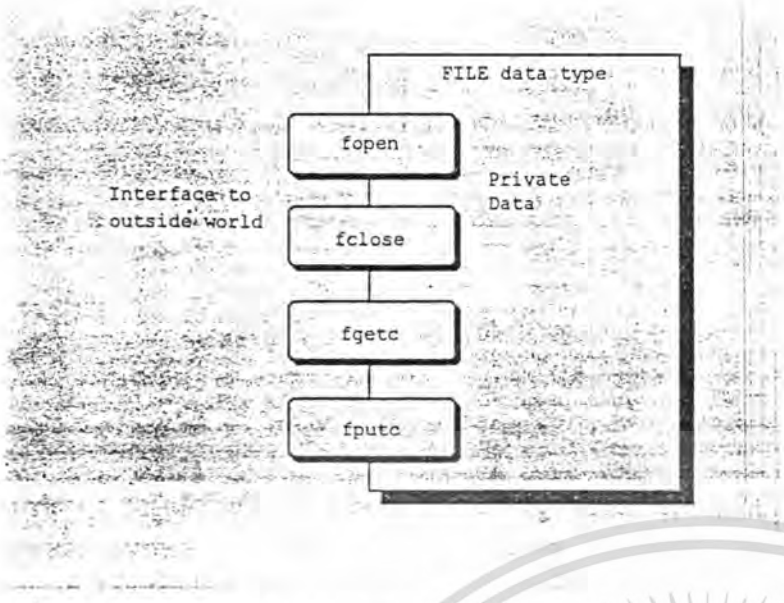
1. ข้อมูลแอบสแทรกชัน
2. อินเฮอริทันซ์
3. โพลิมอร์ฟีซึม

คำศัพท์เหล่านี้ไม่ใช่คำศัพท์ใหม่ แต่การนำมาใช้เป็นรากฐานของโอโอพีเป็นเรื่องใหม่

4.1 ข้อมูลแอบสแทรกชัน

เพื่อทำความเข้าใจกับคำศัพท์ข้อมูลแบบแอบสแทรกชัน พิจารณารูทีนของไฟล์ไอโอ (file I/O routine) ในไลบรารี (library) ของภาษา C ซึ่งรูทีนเหล่านี้มองเห็นไฟล์เป็นเพียงตัวอักษรเรียงติดต่อกันและสามารถถูกกระทำได้หลายวิธีโดยการเรียกใช้รูทีนของไฟล์ไอโอ เช่น fopen เพื่อเปิดไฟล์ fputc เพื่อเขียนข้อมูลในไฟล์ การกำหนดชนิดของข้อมูลเป็น File ซึ่งเก็บข้อมูลเกี่ยวกับไฟล์ ทำให้แบบจำลองนามธรรมของไฟล์เป็นจริงขึ้นมา ในภาษา C ใช้คำสั่ง STRUCT และ TYPEDEF กำหนดข้อมูลชนิดไฟล์

สามารถคิดถึงข้อกำหนดของไฟล์รวมอยู่กับฟังก์ชันที่กระทำข้อมูลนั้น encapsulation) เป็นประเภทของข้อมูลแบบหนึ่งเหมือนเช่นข้อมูลประเภท int



รูปที่ 2 *ข้อมูลชนิด File ในภาษา C ตัวอย่างของเอดีที*

หรือ char การใช้ข้อมูลประเภทไฟล์นั้นไม่จำเป็นต้องรู้โครงสร้างจริง ๆ ของไฟล์ เพราะว่าโครงสร้างของไฟล์อาจเปลี่ยนแปลงไปตามประเภทของระบบคอมพิวเตอร์แต่การทำงานของรูทีนไฟล์ไอโอจะเหมือนกันในทุกระบบ ที่เป็นเช่นนี้ได้เพราะว่าไม่ได้กระทำกับโครงสร้างของข้อมูลโดยตรงแต่ทำผ่านฟังก์ชันและแมคโคร (Macros) ซึ่งได้ซ่อนรายละเอียดภายในของไฟล์ทั้งสิ้นไว้

ข้อมูลแอบสแทรกชันคือการกำหนด ชนิดของข้อมูลรวมทั้งข้อมูลส่วนที่ซ่อนไว้ (เรียกว่า Abstract Data Type หรือ ADT) คำนิยามของเอดีทีคือกำหนดข้อมูลภายในเอดีทีและฟังก์ชันที่กระทำกับเอดีทีนั้น ข้อมูลส่วนที่ซ่อนไว้ทำให้ไม่ต้องเปลี่ยนแปลงโปรแกรมที่เรียกใช้ฟังก์ชันที่กระทำกับเอดีทีนั้นเมื่อมีการเปลี่ยนแปลงโครงสร้างภายในของเอดีที ตัวอย่างของเอดีทีคือ ข้อมูลประเภท File ในภาษา C ดังรูปที่ 2

4.2 ออบเจ็กต์ คลาส เมธธอด

ออบเจ็กต์ในโอโอพีสร้างมาจากเอดีที ซึ่งเอดีทีคือกลุ่มของตัวแปรที่มีฟังก์ชันที่จำเป็นที่กระทำกับตัวแปรเหล่านั้นร่วมอยู่ด้วย ตัวแปรใช้แทนข้อมูลข่าวสารภายในออบเจ็กต์ แต่ฟังก์ชันใช้กำหนดการกระทำที่จะกระทำกับออบเจ็กต์นั้น สามารถคิดถึงเอดีทีเหมือนกับเป็นเทมเพลท (Template) ที่ใช้แทนสมาชิก ของออบเจ็กต์ มักเรียกเทมเพลทว่าคลาส (Class) เพราะฉะนั้นคลาสจึงมีความหมายเหมือนกับเอดีที

ฟังก์ชันที่กระทำกับออบเจ็กต์เรียกว่า Method เป็นคำที่มาจากภาษา Smalltalk ซึ่งเป็นต้นใช้กำหนดความประพฤติของออบเจ็กต์ ในภาษา C++ เรียกว่าเมมเบอร์ฟังก์ชัน (member functions) คำศัพท์คำอื่น ๆ ในโอโอพีมีที่มาจากภาษา Smalltalk แนวความคิดในการเรียกใช้ method คือการส่งข้อความไปยังออบเจ็กต์เพื่อระบุให้ตัวใดตัวหนึ่งใดทำงานในภาษา C++ ก็คือการเรียกเมมเบอร์ฟังก์ชันในออบเจ็กต์

4.3 อินเฮอริแทนซ์

ในการกำหนดข้อมูลชนิดนามธรรมไม่สามารถกำหนดคุณสมบัติที่สำคัญของออบเจ็กต์ได้ครบทั้งหมดและในความเป็นจริงนั้นออบเจ็กต์ไม่ได้แยกกันอยู่แบบเดี่ยว ๆ แต่มีความสัมพันธ์กับออบเจ็กต์อื่น ๆ เสมอ ดังนั้นจึงสามารถที่จะกำหนดออบเจ็กต์ใหม่โดยอธิบายคุณสมบัติที่แตกต่างจากออบเจ็กต์เดิมที่มีอยู่แล้ว ซึ่งเป็นแนวความคิดที่สำคัญของโอโอพีเรียกว่าอินเฮอริแทนซ์ คือการถ่ายทอดข้อมูลและความประพฤติจากคลาสแม่ (Parent class) ไปสู่คลาสลูก (child class) โดยในภาษา C++ เรียกคลาสแม่ว่า เบสคลาส (base class) และเรียกคลาสลูกว่าดิไรว์คลาส (derived class)

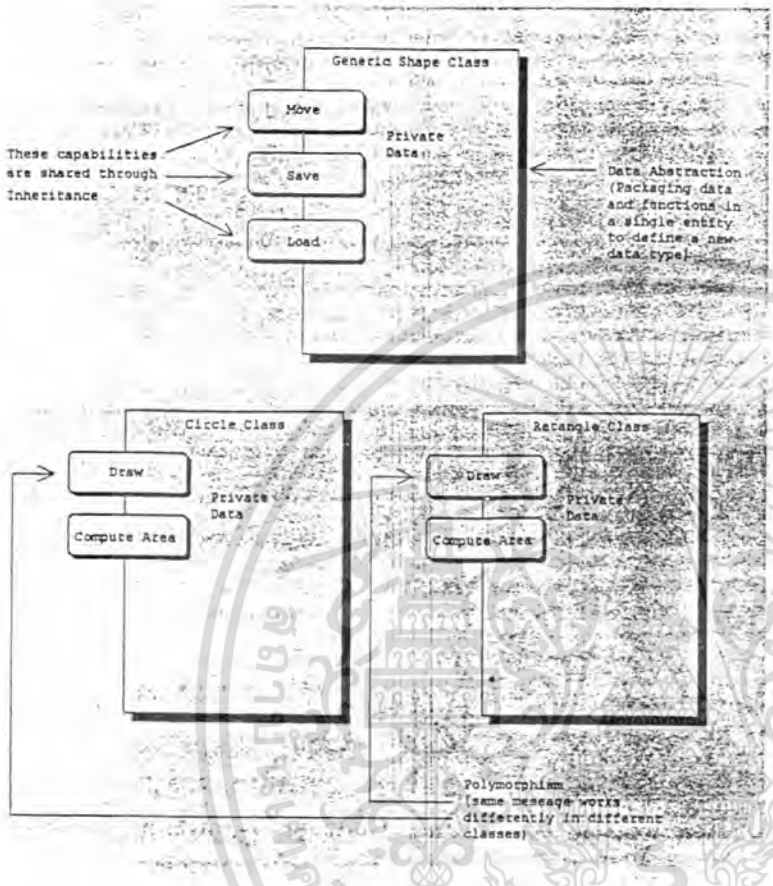
4.4 การอินเฮอริแทนซ์มากกว่าหนึ่ง (Multiple Inheritance)

หมายความว่าดิไรว์คลาสมีการถ่ายทอดมาจากหลาย ๆ เบสคลาสซึ่งในภาษาเชิงวัตถุหลายภาษาไม่สามารถทำได้ แต่ในภาษา C++ สามารถทำได้

4.5 โพลิมอร์ฟิซึ่ม

ความหมายตามคำศัพท์ของโพลิมอร์ฟิซึ่มคือมีคุณสมบัติหลายอย่าง แต่ความหมายโอโอพีหมายถึงการสั่งให้กระทำด้วยคำสั่งเดียวกันกับออบเจ็กต์ที่แตกต่างกันจะมีการกระทำที่แตกต่างกัน เช่น ในตัวอย่างข้างต้นสั่งให้วาดรูปจะได้รูปวงกลมหรือสี่เหลี่ยมขึ้นกับประเภทของออบเจ็กต์นั้น

จากตัวอย่างที่แล้วสามารถเขียนด้วยภาษา C++ โดยใช้เทคนิคของโอโอพี ดังรูปที่ 3



รูปที่ 3 *Data abstraction, inheritance และ Polymorphism ในโอโอพี*

5. ระบบงานบริหารพัสดุสายช่างอากาศ

ในปัจจุบันระบบงานบริหารพัสดุสายช่างอากาศของกรมช่างอากาศควบคุมด้วยระบบคอมพิวเตอร์ที่ใช้โปรแกรมโครงสร้าง ซึ่งบัญหาที่ประสบอยู่เสมอ ๆ คือเมื่อมีการเปลี่ยนแปลงโครงสร้างของข้อมูลจะต้องเปลี่ยนแปลงโปรแกรมทั้งหมดที่เกี่ยวข้องด้วยดังเช่นตัวอย่างที่อธิบายในหัวข้อที่ผ่านมา และต้องการที่จะเพิ่มบางส่วนเข้าไปในระบบงานการดำเนินงานค่อนข้างลำบากยุ่งยากเพราะต้องคำนึงถึงโครงสร้างของแฟ้มข้อมูลเดิม และผลกระทบที่จะเกิดกับโปรแกรมที่ใช้แฟ้มข้อมูลนั้น ดังนั้นจึงได้นำเอาหลักการของการเขียนโปรแกรมเชิงวัตถุมาใช้ในการออกแบบระบบงานบริหารพัสดุช่างอากาศ

5.1 ขั้นตอนของระบบงาน

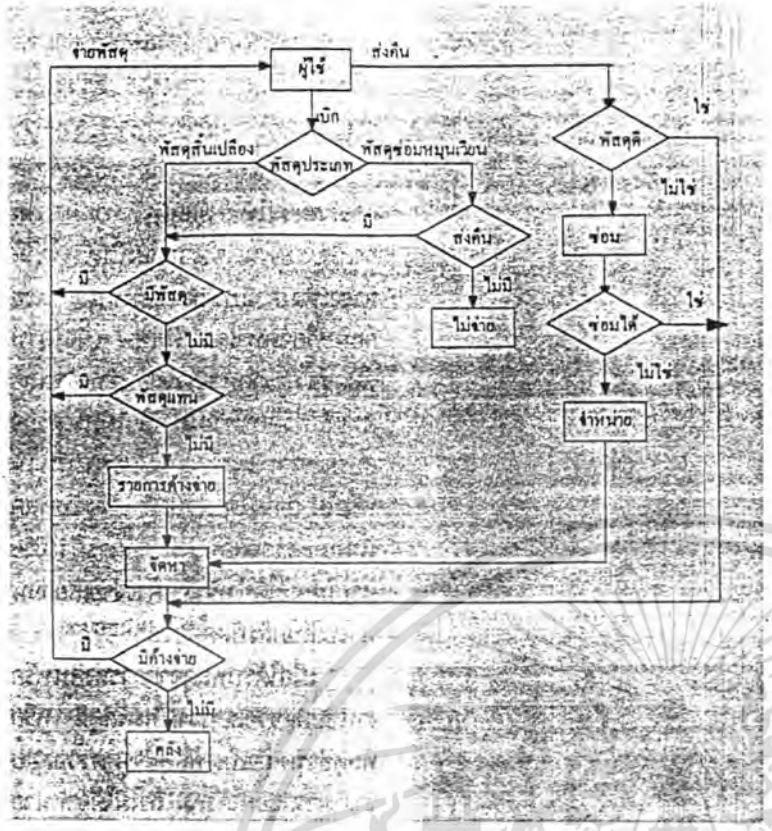
เมื่อมีผู้ใช้เบิกพัสดุ เจ้าหน้าที่พัสดุตรวจสอบประเภทของพัสดุว่าเป็นพัสดุสิ้นเปลือง พัสดุซ่อมหมุนเวียน หรือพัสดที่ต้องมีอัตราจ่าย (เช่น เครื่องมือช่าง) ถ้าเป็นพัสดุซ่อมหมุนเวียนต้องตรวจว่ามีของส่งคืนหรือไม่ เพราะถ้าไม่มีของส่งคืนจะไม่จ่ายให้จากนั้นจึงตรวจสอบว่ามีของจ่ายหรือไม่ ถ้ามีจ่าย ถ้าไม่มีตรวจว่ามีพัสดุที่ใช้แทนกันได้หรือไม่ ถ้าจ่ายพัสดุใช้แทน ถ้าไม่มีทำรายการค้างจ่ายไว้ และพยายามหาหนทางนำพัสดุมาจ่ายเช่น ตรวจสอบรายการรอรับจากการจัดหา รายการรอรับจากงานซ่อม เพื่อที่จะได้เร่งรัดและติดตามให้ได้พัสดุมา หากมีความจำเป็นเร่งด่วน เช่น เครื่องบินบินไม่ได้อาจจะต้องตามขอคืนจากผู้ใช้บางหน่วยก่อนหรือจัดหาโดยวิธีพิเศษ เมื่อได้รับพัสดุจากการจัดหาตรวจสอบว่าหมายเลขตรงกับที่จัดหาหรือไม่ ถ้าไม่ตรงตรวจสอบว่าว่าเป็นพัสดุที่ใช้แทนกันได้หรือไม่ ถ้าไม่ได้ไม่รับ ถ้าได้จึงรับพัสดุนั้นและตรวจสอบว่ามีรายการค้างจ่ายหรือไม่ จะได้ทำหักล้างค้างจ่ายเสียก่อนนำไปเก็บคลังพัสดุต่อไป

เมื่อได้รับพัสดุจากการส่งคืนหรืองานซ่อมตรวจสอบว่าเป็นพัสดุดี หรือพัสดุชำรุด ถ้าเป็นพัสดุชำรุดตรวจสอบว่าสามารถซ่อมได้ไหม ถ้าไม่ได้จำหน่ายพัสดุแล้วจัดหา ถ้าเป็นพัสดุดีตรวจสอบว่ามีรายการค้างจ่ายหรือไม่เพื่อทำหักล้างค้างจ่ายก่อนนำไปเก็บในคลังพัสดุดังแผนผังรูปที่ 4

5.2 ออกแบบระบบงานโดยใช้หลักการของระบบเชิงวัตถุ

จากขั้นตอนการบริหารพัสดุสายช่างอากาศดังที่กล่าวมาข้างต้น ออกแบบระบบงานโดยใช้หลักการของระบบเชิงวัตถุ

กำหนดคลาสประเภทพัสดุ (Genericitem Class) ซึ่งมีข้อมูลรายละเอียดของพัสดุ เช่น หมายเลขพัสดุ ชื่อ จำนวนพัสดุดี จำนวนพัสดุชำรุด รายการรอรับจากการจัดหา (Due In) รายการรอรับจากงานซ่อม (Work Order) รายการค้างจ่าย (Due Out) รายการที่มีการเปลี่ยนแปลงต่าง ๆ (Transaction) เป็นต้น ดังรูปที่ 5



รูปที่ 4 แผนผังระบบงานบริหารพัสดุ

```
Class GenericItem
{Public:
        int     StockNo;
        char    *name;
        int     Onhand;
        int     Repair;
    Collection Duein;
    Collection Workorder;
    Collection Dueout
    Collection Tran;
    Item(int StockNo,char *name);
};
```

รูปที่ 5 การสร้างเบสคลาส

จากรูปที่ 5 ประเภทของข้อมูลที่กำหนดเป็นชนิด Collection นั้น Collection คือคลาสที่ประกอบด้วยข้อมูลที่เป็นคลาสเช่นคลาสของรายการรอรับจากการจัดหา (Duein Class) และ method ดังต่อไปนี้คือ

- เพิ่มข้อมูล (insertAfterCurrent (aNewObject))
- เลื่อนไปที่ข้อมูลถัดไป (object* go To Nex Element())
- แสดงข้อมูลตำแหน่งของบรรทัดปัจจุบัน (object *return Current())
- ลบข้อมูล (deleteCurrent ())
- ไปที่บรรทัดแรก (goToFirstObject ())
- บอกให้รู้ว่าไม่มีข้อมูล (Boolen isEmpty ())

การกำหนดคลาส Collection เป็นข้อมูลส่วนหนึ่งของคลาสประเภทพัสดุ ทำให้สามารถดูรายละเอียดของรายการรอรับจากการจัดหรือจากงานซ่อม รายการค้างจ่าย ของออบเจ็กต์ซึ่งก็คือพัสดุนั่นเองเป็นประโยชน์ในการติดตามสถานะของพัสดุอันเป็นหัวใจของงานบริหารพัสดุที่ต้องรู้ว่าพัสดุมีสถานะเช่นไรมีจำนวนเพียงพอกับภารกิจหรือไม่

```
Class Consumablettem : public GenericItem
{ public :
        virtual int Receive(void);
        virtual int Issue(void);};
Class Reparableltem : public GenericItem
{ public:
        virtual int Receive(void);
        virtual int Issue(void);};
```

รูปที่ 6 การสร้างดิไรว์คลาส

	พัสดุสิ้นเปลือง	พัสดุซ่อมหมุนเวียน
การรับพัสดุ	ตรวจสอบว่ามีรายการค้างจ่ายหรือไม่ - มีทำหักค้างจ่ายที่เหลือเพิ่มจำนวนพัสดุที่มี - ไม่มีเพิ่มจำนวนพัสดุที่มี	ตรวจสอบสภาพพัสดุ - พัสดุดี ตรวจสอบว่ามีรายการค้างจ่ายหรือไม่ ถ้ามีทำหักล้างค้างจ่ายที่เหลือเพิ่มจำนวนพัสดุดี ถ้าไม่มีเพิ่มจำนวนพัสดุดี - พัสดุชำรุด เพิ่มจำนวนพัสดุชำรุด
การจ่ายพัสดุ	ตรวจสอบว่ามีของจ่ายหรือไม่ ถ้ามีจ่าย ไม่มีตรวจสอบว่ามีหมายเลขแทนหรือไม่ ถ้ามีจ่ายหมายเลขแทน ไม่มีทำรายการค้างจ่าย	ตรวจสอบสภาพพัสดุ - พัสดุดี ตรวจสอบว่ามีพัสดุส่งคืนหรือไม่ ถ้าไม่มีไม่จ่าย ถ้ามี และมีของจ่ายจึงจ่าย ถ้าไม่มีของ ตรวจสอบหมายเลขแทนเพื่อจ่ายแทน ถ้าไม่มีหมายเลขแทนทำรายการค้างจ่าย - พัสดุชำรุด ตรวจสอบว่าซ่อมได้หรือไม่ ได้ส่งซ่อม ไม่ได้จำหน่ายพัสดุและจัดหาพัสดุ

รูปที่ 7 ตาราง

แบ่งพัสดุออกตามประเภทของพัสดุ คือเป็นพัสดุสิ้นเปลืองและพัสดุซ่อมหมุนเวียน แล้วกำหนดคลาสพัสดุสิ้นเปลือง (Consumable Item) และคลาสพัสดุซ่อมหมุนเวียน (Reparable Item) เป็นไดรฟ์คลาสสามารถถ่ายทอดคุณสมบัติมาจากเบสคลาสคือ คลาสประเภทพัสดุ มี method การรับพัสดุ (Receive Item) และการจ่ายพัสดุ (Issue-Item) ดังรูปที่ 6 การเรียกใช้ method ในคลาสพัสดุสิ้นเปลือง และคลาสพัสดุซ่อมหมุนเวียนจะมีการกระทำที่แตกต่างกันดังรูปที่ 7

6 สรุป

นอกจากคลาสที่กล่าวมาข้างต้นแล้วยังสามารถกำหนดคลาสเพิ่มขึ้นใหม่ได้อีกตามความจำเป็นโดยไม่ต้องกำหนดเผื่อไว้ล่วงหน้าเช่นวิธีเก่าบางครั้งรุนแรงถึงขั้นต้องรื้อสร้างใหม่ถ้าเผื่อไว้ไม่เพียงพอทำให้ไม่สามารถจะขยายได้ แต่วิธีการของระบบเชิงวัตถุไม่มีปัญหาเช่นนี้ ทำให้การออกแบบระบบทำได้สะดวกและง่าย และสามารถที่จะกำหนดคลาสขึ้นใหม่โดยการอ้างอิงคุณสมบัติของคลาสเดิม และการเพิ่มคลาสใหม่ไม่มีผลกระทบกับส่วนอื่น ๆ ที่มีอยู่เดิมภายในระบบ ทำให้การออกแบบระบบสามารถที่จะเริ่มต้นจากเพียงบางส่วนและทดสอบ เมื่อผ่านการทดสอบแล้วจึงค่อย ๆ ขยายจนครบสมบูรณ์ทั้งระบบ นอกจากนี้สามารถละเว้นส่วนที่ข้อมูลยังไม่สมบูรณ์ไว้ก่อนแล้วมาเพิ่มเติมเข้าไปในระบบภายหลัง สามารถจัดการกับความซับซ้อนของข้อมูลได้ดีและเหมาะกับการออกสายแบบระบบที่มีขนาดใหญ่เช่น ระบบงานบริหารพัสดุช่างอากาศ เป็นต้น

เอกสารอ้างอิง

[1] Bertrand Meyer, 1988, "Object-Oriented Software Construction" (Hertfordshire, Great Britain : Prentice-Hall International UK)

[2] Stroustrup, B., 1986, "The C++ Programming Language", Addison-Weslay.

[3] Rumbaugh, J., Blaha, H. Premerlani, W., Eddy F. and Lorensen, W. 1991,"Object-Oriented Modeling and Design" Prentice Hall, Englewood Cliffs, NJ

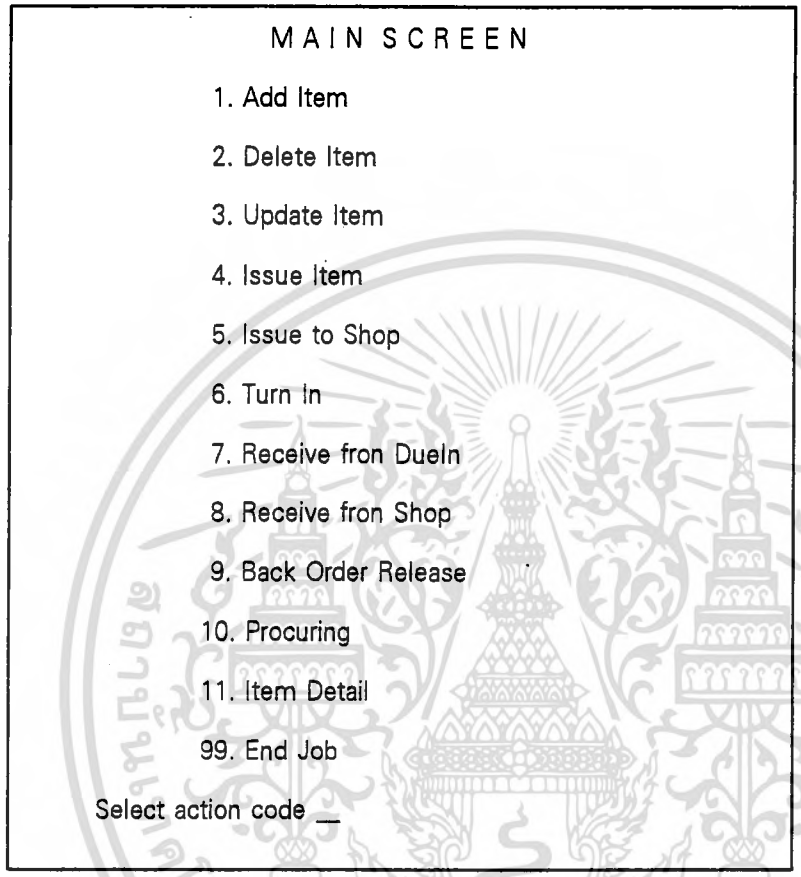
[4] กองทัพอากาศ 2508 "คู่มือว่าด้วยการพัสดุ"

[5] Shlaer, S. and Mellor,S.J., 1988 "Object-Oriented System Analysis Modeling the world in Data" Yourdon Press, Englewood Cliff, NJ.

ภาคผนวก ข
การใช้ระบบงาน

เมนูการใช้งาน

สามารถเลือกประเภทของการทำงานได้จากเมนูหลัก ซึ่งมีรายละเอียดดังนี้



จอภาพของระบบงาน

จอภาพของระบบงานแต่ละจอภาพจะมีลักษณะและวิธีการใช้ที่คล้ายๆ กัน เพื่อให้ผู้ใช้สามารถใช้งานได้ง่าย ไม่ต้องจดจำวิธีการมากมาย โดยที่แต่ละจอภาพจะเริ่มต้นการทำงานด้วยการให้ผู้ใช้ใส่รหัสประเภทของเครื่องบินและหมายเลขพัสดุ โปรแกรมจะนำรายละเอียดของพัสดุนั้นมาแสดงที่จอภาพ จากนั้นผู้ใช้ก็จะใส่รายละเอียดเกี่ยวกับการทำงานในแต่ประเภทที่ต้องการต่อไป จอภาพแบ่งออกเป็น 11 จอภาพคือ

1. การเพิ่มพัสดุใหม่

เริ่มต้นการใช้งานโดยการใส่ประเภทของเครื่องบิน หมายเลขพัสดุ โปรแกรมตรวจสอบว่ามีหมายเลขนี้แล้วหรือยัง เพราะว่าพัสดุหมายเลขเดียวกันสามารถใช้กับเครื่องบินได้หลายประเภท ถ้าพบว่ามีหมายเลขนี้ในเครื่องบินแบบอื่นแล้วจะได้นำเอารายละเอียดของพัสดุได้แก่ ชื่อพัสดุ รหัสหน่วยนับพัสดุ ราคาต่อหน่วย รหัส ERRC มาใช้ได้เลย ผู้ใช้ไม่ต้องเสียเวลาใส่ข้อมูลใหม่อีก และพิมพ์ประเภทของพัสดุว่าเป็นพัสดุสิ้นเปลือง(Consumable) หรือพัสดุซ่อมหมุนเวียน (Reparable) ที่ตำแหน่ง [1] บนจอภาพ แต่ถ้าเป็นพัสดุหมายเลขใหม่ผู้ใช้จะต้องใส่ประเภทของพัสดุและรายละเอียดของพัสดุใหม่ การใส่รหัสแหล่งจัดหาพัสดุ โปรแกรมจะถามว่า more? _ ที่ตำแหน่ง [2] เพราะว่าแหล่งจัดหาพัสดุมีได้มากกว่าหนึ่งแหล่ง ผู้ใช้จะคีย์ตอบด้วยตัวอักขระอะไรก็ได้ในกรณีตอบว่า Yes ทั้งนี้เพื่ออำนวยความสะดวกให้กับผู้ใช้ไม่ต้องเสียเวลาคีย์ตัวอักขระอะไร เพียงแต่กด enter ก็ใส่ข้อมูลต่อไปได้เลย แต่ถ้าต้องการตอบว่า No ต้องใส่คำตอบว่า 'n' ถ้าเป็นหมายเลขที่มีอยู่แล้วจะแสดงแหล่งจัดหาเดิมไว้ให้ด้วย ซึ่งสามารถหรือเพิ่มเติมได้ การใส่หมายเลขพัสดุที่ใช้แทนกันได้ก็มีวิธีการทำเช่นเดียวกันกับการวิธีการใส่รหัสแหล่งจัดหาพัสดุ ดังแสดงรายละเอียดดังนี้

```
                    A D D  I T E M
                    * * * * * * * *

CONSUMABLE [1]
Aircraft type  _              onHand ____
FSC/NIIN or PN _____________
Nomenclature   _____________          ERRC _
Unit Price     _____________          UI __

SOS  ___ ___ ____                                    [2]
ISG  _____________
     _____________
```

2. การลบรายการพัสดุ

กรณีที่มีการจำหน่ายพัสดุจะต้องลบพัสดุนั้นออกไป โปรแกรมจะตรวจสอบก่อนว่ามีรายการรอรับการจัดหาพัสดุหรือรอรับงานซ่อมพัสดุหรือไม่ ถ้ามีโปรแกรมจะไม่ยอมลบพัสดุนั้นให้ ทั้งนี้เพื่อเป็นการป้องกันข้อผิดพลาดที่อาจเกิดขึ้นได้ รายละเอียดดังนี้

```
                    D E L E T E   I T E M
                    * * * * * * * * * * *

CONSUMABLE
Aircraft type  _                onHand ____
FSC/NIIN or PN _____________
Nomenclature   _____________          ERRC _
Unit Price     _____________          UI   __

SOS  ___ ___ ____
ISG  _____________
     _____________

                    Are you sure?(Y/N)__
```

3. การแก้ไขข้อมูล

เมื่อมีความจำเป็นต้องแก้ไขรายละเอียดบางอย่างของพัสดุ รายละเอียดดังนี้

UPDATE ITEM

* * * * * * * * * * * *

CONSUMABLE

Aircraft type _ onHand ____

FSC/NIIN or PN _____

Nomenclature _____ ERRC _

Unit Price _____ UI __

SOS ___ ___ ____

ISG _____

_____

4. การจ่ายพัสดุ

โปรแกรมจะตรวจสอบว่ามีของจ่ายหรือไม่ ถ้าไม่มีจะค้นหาหมายเลขที่ใช้แทนกันได้เพื่อจะจ่ายให้แทน แต่ถ้าไม่มีจะทำรายการค้างจ่ายไว้ รายละเอียดดังนี้

ISSUE ITEM

* * * * * * * * * * *

REPARABLE

Aircraft type _ onHand ____

repair ____

FSC/NIIN or PN _____

Nomenclature _____ ERRC _

Unit Price _____ UI __

Document of Depot _____

Organization _____

Organization Code _____

Quantity _____

5. การจ่ายพัสดุไปซ่อม

การส่งพัสดุชำรุดไปซ่อม ซึ่งจะทำรายการรอรับงานซ่อมเอาไว้ รายละเอียดดังนี้

I S S U E T O S H O P

* * * * * * * * * * * * * *

REPARABLE

Aircraft type _ onHand ____

repair ____

FSC/NIIN or PN _____

Nomenclature _____ ERRC _

Unit Price _____ UI __

Document of Depot _____

Shop _____

Shop Code _____

Quantity _____

Date _____

6. การรับพัสดุส่งคืน

เมื่อผู้ใช้หมดความจำเป็นใช้งานพัสดุจะส่งคืนมายังคลังใหญ่ รายละเอียดดังนี้

T U R N I N

* * * * * * *

REPARABLE

Aircraft type _ onHand ____

repair ____

FSC/NIIN or PN _____

Nomenclature _____ ERRC _

Unit Price _____ UI __

Document of Depot _____

Organization _____

Organization Code _____

Quantity _____

7. การรับพัสดุจัดหา

รายละเอียดของการรับพัสดุจัดหาดังนี้

RECEIVE FROM DUEIN

* * * * * * * * * * * * * * * * * * *

REPARABLE

Aircraft type _ onHand ____

FSC/NIIN or PN _____

Nomenclature _____ ERRC _

Unit Price _____ UI __

Document of Depot _____

Organization _____

Source of Supply Code _____

Quantity _____

Date _____

8. การรับพัสดุจากงานซ่อม

รายละเอียดของการรับพัสดุจัดหาดังนี้

RECEIVE FROM SHOP

* * * * * * * * * * * * * * * * * * *

REPARABLE

Aircraft type _ onHand ____

FSC/NIIN or PN _____

Nomenclature _____ ERRC _

Unit Price _____ UI __

Document of Depot _____

Shop _____

Shop Code _____

Quantity _____

Date _____

9. การทำหักล้างค้างจ่าย

บนจอภาพจะแสดงรายการค้างจ่ายทั้งหมดให้เลือกรายการที่ต้องการทำหักล้างค้างจ่าย ได้ รายละเอียดดังนี้

```
                    B A C K  O R D E R  R E L E A S E
                    * * * * * * * * * * * * * * * * * *

REPARABLE
Aircraft type  _                  onHand ____
                                   repair ____
FSC/NIIN or PN _____________
Nomenclature _____________                ERRC _
Unit Price   _____________                UI   __

                    >>> D U E  O U T <<<

Document
AED Organization    Uid   Quantity   Airtype      Date
__  _________       ___   _____      _____        __________
__  _________       ___   _____      _____        __________
```

[1]

10. การจัดหาพัสดุ

รายละเอียดการจัดหาพัสดุดังนี้

PROCURING

* * * * * * * * * * *

REPARABLE

Aircraft type _ onHand ___

reapir ____

FSC/NIIN or PN _____

Nomenclature _____ ERRC _

Unit Price _____ UI __

Document of Depot _____

Organization _____

Source of Supply Code _____

Quantity _____

Date _____

11. การแสดงรายละเอียดของพัสดุ

รายละเอียดดังนี้

D E T A I L O F I T E M

* * * * * * * * * * * * * *

REPARABLE

Aircraft type _ onHand ____

repair ____

FSC/NIIN or PN _____

Nomenclature _____ ERRC _

Unit Price _____ UI __

> > > D U E O U T < < <

___ _____ __ _____ _____ _____

___ _____ __ _____ _____ _____

> > > D U E I N < < <

___ _____ __ _____ _____ _____

___ _____ __ _____ _____ _____

> > > W O < < <

___ _____ __ _____ _____ _____

___ _____ __ _____ _____ _____

ภาคผนวก ค
ตัวอย่างรายงาน

LISTING OF ITEM

AIRCRAFT	FSC/NIIN	NOMENCLATURE	PRICE	UI	ON-HAND	REPAIR	DUO	DUI	WO

แสดงรายการของพัสดุทั้งหมด แยกตามประเภทของเครื่อง
เรียงลำดับตามหมายเลขพัสดุ

D E T A I L O F I T E M

* * * * * * * * * * * * * *

REPARABLE

Aircraft type _ onHand ____

repair ____

FSC/NIIN or PN _____

Nomenclature _____ ERRC _

Unit Price _____ UI __

> > > D U E O U T < < <

__ _____ __ _____ _____ _____

__ _____ __ _____ _____ _____

> > > D U E I N < < <

__ _____ __ _____ _____ _____

__ _____ __ _____ _____ _____

> > > W O < < <

__ _____ __ _____ _____ _____

__ _____ __ _____ _____ _____

LISTING OF DUE IN

DOCUMENT	DATE	FSC/NIIN	NOMENCLATURE	PRICE	UI	QUANTITY

แสดงรายการรอรับการจัดหาพัสดุทั้งหมด

เรียงลำดับตามหมายเลขเอกสารการจัดหา

เอกสารอ้างอิง

[1] Bertrand Meyer,Object-Oriented Software Construction,Hertferdshire,Great Britain:Prentice hall International, UK, 1988

[2] Rambaugh, J.,Blaha, H.,Premerlani, W.,Eddy F, and Lorensen, W. , Object-Oriented Modeling and Design, Prentice Hall, Englewood Cliffs,New Jersey, 1991

[3] Digitalk Inc., Smalltalk/v Tutorial and Programming Handbook,Digitalk Inc.,1986

[4] Shlaer, s., and Mellor, s.J., Object-Oriented System Analysis: Modeling the word in Data, Prentice Hall, Englewood Cliffs,New Jersey, 1991

[5] Peter Coad and Edward Yourdon, Object-Oriented Analysis, Prentice Hall, Englewood Cliffs,New Jersey, 1991

[6] Embley, w.,Kurtz, d., and Woodfield, n., Object-Oriented Systems Analysis A Model-Driven Approach, Prentice Hall, Englewood Cliffs,New Jersey, 1992

[7] กองทัพอากาศ, คู่มือว่าด้วยการพัสดุ, โรงพิมพ์กองทัพอากาศ, กรุงเทพฯ 2508

[8] Mark Mullin, Object-Oriented Program Design with Example in c++, Addison-Wesley Publishing Company,Inc., 1989

[9] J. g. Hughes, Object-Oriented Database, Prentice hall International, UK, 1991

[10] Cardenas, f., and Mcleod, Research Foundations in Object-Oriented and Semantic Database Systems

ประวัติผู้เขียน

ชื่อผู้เขียน	น.ท.หญิง นพพร สงวนทรัพย์
วันเดือนปีเกิด	22 เมษายน 2499
สถานที่เกิด	จ.ปทุมธานี
วุติการศึกษาระดับปริญญาตรี	วท.บ.(คณิตศาสตร์)
สถานที่สำเร็จการศึกษา	คณะวิทยาศาสตร์ มหาวิทยาลัยเกษตรศาสตร์
ปีที่สำเร็จการศึกษา	ปีการศึกษา 2522
ผลงานวิชาการ	วารสารสมาคมคอมพิวเตอร์ "การใช้โปรแกรมเชิงวัตถุกับระบบงานบริหารพัสดุสายช่างอากาศ"
สถานที่ทำงาน	กรมการสนเทศทหาร กองบังชาการทหารสูงสุด